Universidad de Buenos Aires
Facultad de Ciencias Exactas y Naturales
Departamento de Computación

Traducciones entre λ-cálculos con patrones

Lucio Santi
lsanti@dc.uba.ar

Director
Dr. Ariel Arbiser
FCEyN UBA

Tesis de
Licenciatura en Ciencias de la Computación

19 de junio de 2009

Resumen

Nuestro trabajo estudia relaciones entre distintos λ-cálculos con patrones a través de la definición de traducciones entre ellos, a nivel sintáctico. Presentamos la traducción de un gran fragmento del cálculo lambda con patrones múltiples ($\lambda\mathcal{C}$) al cálculo lambda con constructores ($\lambda\mathcal{B}_c$). Esto tiene como fin la posibilidad de compilación de un lenguaje de características y operaciones complejas que están "internalizadas", como el primero, en un lenguaje con un sistema de patrones minimales dados por el análisis por casos sobre constantes básicas, como el segundo, que incluye a cambio un conjunto de reglas de propagación de este constructor de análisis por casos. Tenemos también interés en codificar con combinadores ciertos cálculos con patrones. Para esto, presentamos una formulación de un cálculo de combinadores para $\lambda\mathcal{B}_c$. Si bien los combinadores **S** y **K** clásicos de la lógica combinatoria son funcionalmente completos (en el sentido de que permiten expresar todos los términos del cálculo lambda), proponemos una extensión de esta lógica a través de otros combinadores posibles para la propagación del constructor del análisis de casos, con el fin de obtener un sistema funcionalmente completo y minimal de combinadores para $\lambda\mathcal{B}_c$. Así, se provee un mecanismo de implementación del pattern matching del mismo modo que la lógica combinatoria clásica provee una implementación del cálculo lambda. Para este nuevo sistema probamos su capacidad de abstracción y la confluencia (la cual garantiza la unicidad de formas normales).

Abstract

Our work studies relations between different λ-calculi with patterns by means of translations between them, to the syntactical level. We present the translation of a big fragment of the λ-calculus with multiple patterns ($\lambda\mathcal{C}$) to the λ-calculus with constructors ($\lambda\mathcal{B}_c$). This has as goal to make it possible to compile a language of complex characteristics and operations which are internalized, like the former, into another with a minimal pattern system given by the case construct over basic constants, like the latter, which in turn includes a set of rules for propagating this case construct. We are also interested in coding with combinators certain pattern calculi. For this task we present a formulation of a combinator calculus for $\lambda\mathcal{B}_c$. Although the classical combinators **S** and **K** of combinatory logic are functionally complete (in the sense that they can represent all the λ-calculus terms), we propose an extension of this logic by means of other possible combinators for handling the propagation of the case construct, the goal being to obtain a minimal functionally complete system of combinators for $\lambda\mathcal{B}_c$. Therefore, a mechanism for implementing pattern matching is given, much in the same way as classical combinatory logic provides an implementation of λ-calculus. For this new system we prove its capability of abstraction and its confluence (which guarantees the uniqueness of normal forms).

Agradecimientos

En primer lugar, quiero agradecer a Ariel por su responsable dirección y por todo el tiempo invertido a lo largo del desarrollo del trabajo. También doy las gracias a los jurados por la lectura y revisión de la tesis. Quiero además agradecer a mi familia por el apoyo en todo momento y el afecto transmitido (mención especial para mi hermano Leandro, por los siempre bienvenidos tips de LaTeX), y, para terminar, a todos mis amigos por la compañía y los buenos momentos que pasamos juntos (mención especial para Germán, David y Daniel, excelso grupo de trabajos prácticos durante toda la carrera, gracias a quienes puedo hoy estar en esta posición).

Índice general

Índice de figuras

Capítulo 1

Introducción

En las ciencias de la computación, la teoría de la reescritura [8, 13] estudia distintas formas de transformaciones sintácticas entre expresiones de diversos lenguajes formales. Estrechamente relacionada con éstos y con la lógica matemática, surge como área independiente en 1972 a partir del trabajo de Knuth y Bendix [46] sobre la decidibilidad de determinar la confluencia de ciertos sistemas de reescritura de términos mediante el análisis de los llamados pares críticos [44, 8, 13, 52, 53, 57]. La reescritura es una vasta área, muy activa en diversas universidades del mundo desde hace unos 25 años, y actualmente permite distintas formulaciones, desde las totalmente abstractas hasta las muy concretas [62, 50, 8, 47, 17, 13], por ejemplo las basadas en signaturas de primer orden [8, 47, 13], y en el cálculo lambda bajo sus distintas variantes [18, 19, 9, 10, 13]. Las aplicaciones de la reescritura son variadas: a la programación, a la lógica, semántica de lenguajes, teoría de tipos, etc. El cálculo lambda (o λ-cálculo), creado por Alonzo Church en 1936 [18, 19], pasó a ser un caso particular de sistema de reescritura, y en el que hoy en día se basa la programación funcional.

Actualmente, existen por lo menos tres líneas principales en el estudio de reescritura: los sistemas abstractos de reescritura [8, 13, 50], para los cuales se trata de hallar nuevas técnicas de prueba de confluencia y normalización; el cálculo lambda, del cual se intenta proponer nuevas variantes a partir de distintos problemas que surgen; y los sistemas de reescritura de orden superior [44, 39, 52, 53, 56, 57, 62, 34, 14], que sirven de escenario unificado para estudiar diversos formalismos con distintos poderes expresivos.

El cálculo lambda resultó ser un lenguaje de programación universal en el que las funciones son, como se suele decir, "ciudadanos de primera clase": se puede aplicar una función a otra función y se puede devolver una función como resultado de otra. A pesar de la simplicidad de su sintaxis, este lenguaje es suficientemente rico para representar todas las funciones computables. Desde los orígenes de las ciencias de la computación, el cálculo lambda ha sido utilizado exitosamente ya que constituye el núcleo de los lenguajes de programación funcional, desde LISP hasta los lenguajes de la familia ML o cercanos (como Caml, SML y Haskell). Resulta también de interés el estudio de los sistemas tipados [10]. En virtud de ello, el cálculo lambda es una herramienta fundamental para describir el comportamiento de las demostraciones matemáticas a través de la correspondencia de Curry-Howard [65], por lo cual resulta ser la base de los asistentes de prueba usados en la actualidad (Coq, Nuprl, Isabelle).

Como una formulación alternativa del cálculo lambda surge la lógica combinatoria [63, 20, 21, 22, 23], que es un sistema de reescritura en cierto sentido más simple pero sin embargo igualmente expresivo, permitiendo simular el mecanismo de abstracción de aquél y representar todas las funciones computables. Se presenta dentro del esquema de un sistema

de reescritura de términos de primer orden, lo cual da una mayor simpleza de representación y estudio, así como para posibles implementaciones, puesto que no utiliza el concepto de variable ligada.

Aunque el cálculo lambda puede representar cualquier estructura de datos, los lenguajes de programación y los asistentes de prueba modernos extienden el lenguaje básico con construcciones primitivas para representar estas estructuras con el fin de evitar la ineficiencia inducida por una codificación puramente funcional, en general de difícil utilización y costosa en términos de recursos. Una de estas extensiones es el mecanismo de filtrado de valores (*pattern matching*, o coincidencia de patrones), problema al que se le ha dedicado mucha atención en la programación funcional u orientada a objetos, en diferentes cálculos con patrones [58, 16, 36, 15, 29, 43, 30, 31] y en formalismos de reescritura de orden superior. Un patrón es esencialmente una especificación sintáctica de una familia de argumentos posibles, que facilita las definiciones por casos, lo cual es práctica corriente en la programación declarativa actual [60]. Los patrones son de gran interés en programación funcional, ya que permiten definir funciones en forma concisa y en general de más fácil comprensión y mantenimiento para un programador. El pattern matching permite verificar si un dato de entrada se ajusta a -aparea, coincide, encaja o se corresponde con- un patrón dado expresado en el código del programa.

1.1. Objetivos de esta tesis

Nuestro trabajo se enmarca principalmente en lenguajes con mecanismos de filtrado de valores, más precisamente aquellas variantes del cálculo lambda que incorporan el manejo de patrones. En particular son de interés el λ-cálculo con patrones [58, 45], el *Pure Pattern Calculus* [43], el cálculo lambda con constructores [3, 5, 6] y los *Pure Pattern Type Systems* [11, 68]. El cálculo lambda con patrones es una extensión mínima y natural del cálculo lambda clásico en donde se implementa la noción de pattern matching de manera implícita e instantánea. El Pure Pattern Calculus es un formalismo con patrones donde éstos son ciudadanos de primera clase: se puede utilizar un patrón como argumento de una función y se puede devolver un patrón como resultado de una función. El cálculo lambda con constructores combina constructores con funciones. Los Pure Pattern Type Systems son una extensión con patrones de los *Pure Type Systems* [10], gracias a los cuales se puede definir una jerarquía de cálculos tipados de manera general.

La mayoría de estos cálculos han sido formulados sobre lenguajes sin tipos con la finalidad de estudiar patrones en su estado más puro. Es nuestro interés conocer cómo se relacionan los distintos formalismos a través del manejo que hacen de los patrones. Esto motiva la necesidad de estudiar mapeos y traducciones entre las distintas formulaciones de cálculos con patrones, en uno u otro sentido.

Nuestro trabajo principal será la traducción de un gran fragmento del cálculo lambda con patrones múltiples ($\lambda\mathcal{C}$) al cálculo lambda con constructores ($\lambda\mathcal{B}_c$). Esto tiene como fin la posibilidad de una compilación de un lenguaje de características y operaciones complejas que están "internalizadas", como el primero, en un lenguaje con un sistema de patrones minimales dados por el análisis de casos sobre constantes, teniendo como única operación de apareo la igualdad de éstas, junto a otras reglas de propagación de estos patrones que resultan necesarias dada la sencillez de éstos.

Tenemos también interés en encontrar distintas formulaciones de un cálculo con combinadores para los diversos cálculos con patrones o constructores. Si bien los combinadores **S** y **K** clásicos de la lógica combinatoria son funcionalmente completos (en el sentido de que permiten expresar todos los términos del λ-cálculo y representar todas las funciones

computables), propondremos una extensión de esta lógica definiendo un sistema de combinadores que generalizan **S** y **K** con la incorporación de pattern matching, con el fin de obtener un sistema funcionalmente completo y minimal de combinadores para el λ-cálculo con constructores. Así, el contar con un adecuado cálculo de combinadores proveería mecanismos de implementación del pattern matching del mismo modo que la lógica combinatoria clásica provee una implementación del cálculo lambda. Sobre este nuevo sistema interesará probar la simulación y la confluencia (la cual garantiza la unicidad de formas normales, cuando éstas existen).

Transformaciones sintácticas entre distintos cálculos con patrones y constructores, como las mencionadas, permitirán transferir propiedades de manera natural de un formalismo hacia otro, y comprender mejor la relación que hay entre estos cálculos, un aspecto que no llega a estar del todo resuelto en la actualidad.

1.1.1. Plan de la tesis

En el capítulo 2 definiremos y analizaremos distintas propiedades de una traducción que permitirá convertir términos de $\lambda\mathcal{C}$ en términos de $\lambda\mathcal{B}_c$. Luego, el capítulo 3 presentará una sistema de combinadores para $\lambda\mathcal{B}_c$ junto con sus propiedades esenciales (confluencia y representación de la abstracción), así como también traducciones en ambos sentidos entre $\lambda\mathcal{B}_c$ y dicho sistema. Por último, en el capítulo 4 expondremos las conclusiones generales y las líneas de trabajo futuro.

En el apéndice A, además, comentaremos brevemente las características de nuestra implementación de cada cálculo estudiado y de las distintas traducciones desarrolladas.

1.2. Preliminares

El objetivo de esta sección es contextualizar al lector dándole los elementos que precisará conocer y manejar a lo largo de todo el documento. Para ello, describiremos brevemente los distintos formalismos involucrados en nuestro desarrollo. Para más información y para obtener acceso a las pruebas involucradas, se aconseja consultar la bibiliografía sugerida en cada caso.

1.2.1. Reescritura básica

En primer lugar, repasaremos algunas definiciones clásicas de la teoría de reescritura (recurrir a [8, 13] para un tratamiento más profundo del tema).

Definición 1.2.1. Un *Sistema Abstracto de Reescritura (ARS)* es un par $A = (|A|, \to_A)$ formado por un conjunto arbitrario $|A|$ (llamado *carrier set* de A) junto con una relación binaria $\to_A$ sobre $|A|$. Denotamos mediante $\to_A^*$ (o, alternativamente, $\twoheadrightarrow_A$) la clausura reflexiva-transitiva de $\to_A$ (es decir la menor relación reflexiva y transitiva que incluye a $\to_A$), y mediante $\to_A^=$, la clausura reflexiva de $\to_A$ (es decir la menor relación reflexiva que incluye a $\to_A$).

Usualmente, dado un ARS A, llamaremos *reducción* a la relación $\to_A$.

Definición 1.2.2. Un ARS A es *fuertemente normalizante* (SN) si no existe una secuencia infinita de objetos $(M_i)_{i\in\mathbb{N}} \in |A|^{\mathbb{N}}$ tales que $M_i \to_A M_{i+1}$ para cada $i \in \mathbb{N}$.

Definición 1.2.3. Sean $A = (S, \to_A)$ y $B = (S, \to_B)$ dos ARSs definidos sobre el mismo carrier set S. Decimos que:

- *A conmuta débilmente* con B, escrito $A \,/\!/_w\, B$, si para M, M_1, M_2 cualesquiera tales que $M \to_A M_1$ y $M \to_B M_2$, existe M_3 tal que $M_1 \to_B^* M_3$ y $M_2 \to_A^* M_3$.
- *A conmuta* con B, escrito $A \,/\!/\, B$, si para M, M_1, M_2 cualesquiera tales que $M \to_A^* M_1$ y $M \to_B^* M_2$, existe M_3 tal que $M_1 \to_B^* M_3$ y $M_2 \to_A^* M_3$.

Un ARS A se dice *débilmente confluente* o *weakly Church-Rosser (WCR)* (respectivamente *confluente*, o *Church-Rosser (CR)*) sii $A \,/\!/_w\, A$ (respectivamente sii $A \,/\!/\, A$).

Además, se dice que A satisface la *propiedad diamante* (notada $\diamond$) si, para M, M_1, M_2 cualesquiera tales que $M \to_A M_1$ y $M \to_A M_2$, existe M_3 tal que $M_1 \to_A M_3$ y $M_2 \to_A M_3$.

1.2.2. El λ-cálculo clásico

El λ-cálculo fue introducido por Church en los años 30 [19] como un lenguaje universal para expresar computaciones de funciones. A pesar de su notable simplicidad, es lo suficientemente expresivo como para representar todas las funciones computables. Este formalismo ha sentado las bases de los lenguajes de programación funcionales, desde LISP hasta auqéllos de la familia de ML. En lógica, el λ-cálculo constituye hoy en día una herramienta fundamental para describir los contenidos computacionales de las pruebas a través de la correspondencia de Curry-Howard [65].

1.2.2.1. Sintaxis

El λ-cálculo posee dos operaciones básicas: *aplicación* (notada $F\,M$, donde F, considerado como función, se aplica a M, considerado como argumento) y *abstracción* (notada $\lambda x.M$, que representa un mapeo $x \mapsto M$). De esta manera, el conjunto de λ-términos Λ puede definirse, en sintaxis abstracta,

$$M, N \;::=\; x \mid M\,N \mid \lambda x.M$$

siendo x una *variable* de un conjunto infinito numerable de variables.

1.2.2.2. Variables ligadas y α-conversión

La variable x se dice *ligada* en el término $\lambda x.M$. Una variable que tiene al menos una ocurrencia no ligada en un término M cualquiera, se dice *libre* en M. El conjunto de variables libres de un término M cualquiera se nota $\mathrm{FV}(M)$.

El renombre de variables ligadas de un término no altera su comportamiento. Tal renombre es llamado *α-conversión*. La *convención de variables de Barendregt* permite trabajar con términos donde los nombres de las variables ligadas serán distintos a los nombres de las variables libres.

1.2.2.3. β-reducción y η-reducción

La regla básica de reducción en Λ es la llamada β-reducción:

$$(\lambda x.M)\,N \to_\beta M[x \,/\, N]$$

donde $M[x \,/\, N]$ es la sustitución en M de cada ocurrencia libre de x por el término N.

Por otro lado, la η-reducción es útil para eliminar abstracciones redundantes. Si el propósito de una abstracción sólo es pasar en forma directa su argumento a otra función, entonces tal abstracción es redundante y puede extraerse a través de una reducción η:

$$\lambda x.M\,x \to_\eta M$$

donde $x \notin \mathrm{FV}(M)$.

Se dice que M es una forma (β-)normal si no existe N tal que $M \to_\beta N$. Las formas normales representan, como en todo sistema de reescritura, el resultado final del cómputo. No todo término tiene forma normal: por ejemplo,

$$(\lambda x.xx)\,(\lambda x.xx) \to_\beta (\lambda x.xx)(\lambda x.xx) \to_\beta \cdots$$

donde se ve que no hay posiblilidad de alcanzar un término que no admita reducciones.

Desde un punto de vista teórico, el λ-cálculo goza de diversas propiedades, entre las cuales cabe destacar la confluencia de la β-reducción (lo que implica el determinismo de cómputos): si $M \twoheadrightarrow_\beta M_1$ y $M \twoheadrightarrow_\beta M_2$ entonces existe M_3 tal que $M_1 \twoheadrightarrow_\beta M_3$ y $M_2 \twoheadrightarrow_\beta M_3$. En particular, el hecho de que valga la confluencia implica que hay unicidad de formas normales; en otras palabras, un término dado no puede tener más que una sola forma normal.

1.2.3. Lógica combinatoria

La lógica combinatoria (CL) es un mecanismo introducido por Curry y Schönfinkel alrededor de la década del '30, cuyo objetivo inicial era el de eliminar la necesidad de variables cuantificadas en la lógica matemática. No obstante, su uso más popular recae en las ciencias de la computación, en donde es utilizada como un modelo computacional debido a que, a pesar de su simplicidad, logra capturar distintos aspectos de las computaciones.

Este formalismo puede entenderse como una variante del λ-cálculo, en donde las abstracciones y las variables ligadas son eliminadas para dar lugar a constantes funcionales llamadas *combinadores*. El corazón del sistema consiste en un conjunto inicial de combinadores y la aplicación entre ellos (siguiendo determinadas reglas) para generar nuevos combinadores. Los combinadores iniciales son llamados **K** y **S**. Si bien existen distintas interpretaciones de estos combinadores, ya sea dentro de la reescritura misma así como también en la versión lógica (i.e., isomorfismo de Curry-Howard), informalmente, el combinador **K** sirve para generar funciones constantes, mientras que **S** consiste en una versión generalizada de la aplicación.

1.2.3.1. Sintaxis

Dado un conjunto infinito numerable de variables (notadas x, y, z, etc.), el conjunto de términos **CL** de la lógica combinatoria (notados con M, N, etc.) puede definirse, en sintaxis abstracta,

$$M, N ::= x \mid \mathbf{K} \mid \mathbf{S} \mid M\,N$$

1.2.3.2. Reglas de reducción

El interés sobre la lógica combinatoria viene dado por las reglas de reducción asociadas a **K** y **S**. El sistema posee únicamente las siguientes reglas de reducción:

$$\begin{array}{rcl} \mathbf{K}\,x\,y & \to & x \\ \mathbf{S}\,x\,y\,z & \to & x\,z\,(y\,z) \end{array}$$

La intuición detrás de la regla de **S** es que permite aplicar x a y, pero luego de sustituir a z sobre cada uno de ellos.

La reducción $\to_{\mathrm{CL}}$ denota la unión de ambas reglas.

1.2.3.3. Propiedades

Puede demostrarse que la reducción $\to_{\mathrm{CL}}$ es confluente [13]. En relación con el vínculo entre el λ-cálculo y la lógica combinatoria, es posible definir una forma de representar la abstracción en CL, notada λ^*. Tal representación verifica la siguiente proposición:

Proposición 1.2.1. Sean $M, N \in \mathbf{CL}$. Luego,

$$(\lambda^* x.M)\, N \twoheadrightarrow_{\mathrm{CL}} M[x \,/\, N]$$

siendo $M[x \,/\, N]$ la sustitución de variables definida de la manera esperada.

A partir de esto, es posible definir mapeos del λ-cálculo hacia CL (notado $(\bullet)_{\mathrm{CL}}$) y viceversa (notado $(\bullet)_\lambda$). Este último mapeo es tal que se verifica la

Proposición 1.2.2. Sean $M, N \in \mathbf{CL}$ tales que $M \to_{\mathrm{CL}} N$. Luego,

$$M_\lambda \to_\beta^+ N_\lambda$$

1.2.4. Un cálculo con patrones múltiples: $\lambda\mathcal{C}$

El λ-cálculo con patrones surgió a partir de la práctica en el diseño de lenguajes de programación funcionales. Esencialmente, $\lambda\mathcal{C}$ se trata de un cálculo con constructores (i.e., constantes con una determinada aridad asociada) y pattern matching básico, en el cual la sintaxis del λ-cálculo original es extendida con constructores y términos construidos a partir de ellos [45]. Estos constructores son rígidos e inmutables; no existen reglas de reducción asociadas a ellos. Los patrones se generan en base a los constructores y las variables: un patrón es un término de primer orden construido a partir de dichos objetos. Es de especial importancia el estudio de los patrones *lineales*. Se dice que un patrón es lineal cuando ninguna variable ocurre más de una vez en él.

1.2.4.1. Sintaxis

Siendo Var un conjunto infinito numerable de variables (notadas x, y, z, etc.) y $\mathcal{C}$ un conjunto enumerable de constructores (notados c, c', etc., y con una aridad δ asociada) tales que Var y $\mathcal{C}$ son disjuntos, el conjunto de términos $\Lambda\mathcal{C}$ (M, M_1, etc.) y el conjunto de patrones $\mathcal{P}$ (P, P_1, etc.) pueden definirse mediante las siguientes gramáticas:

Términos	M	::=	x	(Variable)
		|	$c(M_1, \cdots, M_{\delta(c)})$	(Constructor con argumentos)
		|	$M_1\, M_2$	(Aplicación)
		|	$\lambda P_1.M_1 \,\vert\, \cdots \,\vert\, \lambda P_k.M_k$	(Abstracción múltiple)
Patrones	P	::=	x	(Variable)
		|	$c(P_1, \cdots, P_{\delta(c)})$	(Constructor con argumentos)

Como puede verse, existen términos que consisten en constructores con sus respectivos argumentos, donde cada uno de éstos puede ser un término arbitrario a diferencia de los patrones, para los cuales cada argumento debe ser, a su vez, un patrón.

1.2.4.1.1. Abstracciones múltiples sobre patrones

La característica distintiva de $\lambda\mathcal{C}$ es la generalización de la abstracción clásica: a diferencia del λ-cálculo original, no se abstrae sobre variables sino sobre patrones. Además de esto, como se ve en la definición de la sintaxis, el cálculo introduce abstracciones *múltiples*: una abstracción es ahora una construcción con una cantidad arbitraria (finita) de patrones abstraidos junto con los respectivos cuerpos de las funciones asociadas.

1.2.4.1.2. Variables libres y ligadas

Observar que, en este caso, las variables ligadas de un término cualquiera deben definirse teniendo en cuenta que, en una abstracción múltiple, un patrón puede ligar más de una variable. A partir de esto, el conjunto de variables libres de un término M, $\mathrm{FV}(M)$, se define de la manera esperada.

1.2.4.2. Reducción β-multiple-pattern

La evaluación en $\lambda\mathcal{C}$ se da a través de la regla de reducción β-*multiple-pattern*, que puede entenderse como una generalización de la β-reducción clásica:

$$(\lambda P_1.M_1 \mid \cdots \mid \lambda P_k.M_k)\, P_i^\sigma \rightarrow_{\beta\mathrm{p}} M_i^\sigma$$

donde σ es una sustitución (i.e., un mapeo finito de variables a términos) que opera sobre las variables del patrón P_i, y $1 \leq i \leq k$. Notar que un redex βp debe ser tal que el argumento de la abstracción múltiple sea una *instancia* de algún patrón en dicha abstracción. De no existir tal sustitución σ, simplemente no podrá realizarse el paso de reducción (i.e., no existirá el redex).

La reducción de patrones en $\lambda\mathcal{C}$ permite acelerar las reducciones respecto del λ-cálculo clásico: allí, las estructuras de datos deben ser codificadas explícitamente mientras que, en este caso, el pattern matching es implícito, otorgando una metodología más natural para expresar operaciones sobre tipos de datos.

1.2.4.3. Confluencia de $\lambda\mathcal{C}$ y linealidad

La relación entre confluencia y linealidad de los patrones es de gran importancia. En primer lugar, los autores, en [45], demuestran que, de tener patrones no lineales, la reducción β-multiple-pattern no necesariamente es confluente. Sin embargo, tampoco alcanza únicamente con linealidad para poder asegurar confluencia. El problema ocurre si un argumento de una abstracción múltiple instancia más de un patrón. Prohibiendo también esta condición, puede garantizarse la confluencia, según lo enuncia el siguiente

Teorema 1.2.1. Para patrones lineales y no unificables de a pares, la reducción βp es confluente.

La prueba de 1.2.1 se basa en el concepto de ortogonalidad [13].

Es de destacar que $\lambda\mathcal{C}$ resulta un cálculo significativo pues extiende a una versión del cálculo λP [58] (a su vez restringido a patrones de primer orden sin contemplar abstracciones). λP fue el primer cálculo que, extendiendo al λ-cálculo clásico, incluyó la posibilidad de utilizar patrones en las abstracciones.

1.2.5. Un cálculo con constructores: $\lambda\mathcal{B}_\mathcal{C}$

$\lambda\mathcal{B}_c$ es una extensión del λ-cálculo clásico que incorpora constructores y análisis de casos, con el objetivo de simplificar la codificación de distintas estructuras de datos manteniendo un marco formal para el estudio matemático de los lenguajes de programación funcionales. Su principal novedad es la conmutación entre la aplicación y el análisis de casos, a través de la regla [1]

(CaseApp) $\qquad\qquad \theta.(M\,N) \;\rightarrow\; \theta.M\,N$

[1] Observar que M se trata como a una función en el lado izquierdo de la regla, mientras que se trata como a un valor construido del lado derecho.

donde θ representa un *case binder*: un mapeo finito de constructores a términos. De igual manera, existe una regla

$$\text{(CaseLam)} \qquad \theta.(\lambda x\,.\,M) \;\to\; \lambda x\,.\,(\theta.M) \qquad (x \notin \mathrm{FV}(\theta))$$

que permite al análisis de casos penetrar las abstracciones.

El sistema obtenido no sólo es computacionalmente correcto (es confluente y conservativo sobre el $\lambda\eta$-cálculo), sino que además permite descomponer el *pattern matching* (al estilo ML, con patrones de aridad arbitraria) mediante la construcción $\theta.M$ que realiza análisis de casos únicamente en constructores constantes (ver la sintaxis en la próxima sección). La prueba de confluencia de este cálculo se apoya en resultados clásicos de reescritura y otros lemas.

Un teorema de separación débil para el cálculo completo también se probó en [5], utilizando una técnica de separación inspirada por el trabajo de Böhm [9]. Por esta razón, el formalismo provee una constante especial, *daimon*, notada ✠, que solicita la terminación del programa y que se utiliza como mecanismo técnico principal para observar formas normales y separarlas.

Las pruebas y demás aspectos relevantes sobre esta información preliminar están disponibles en [5].

1.2.5.1. Sintaxis

Este cálculo distingue, al igual que $\lambda\mathcal{C}$, dos tipos de nombres: variables (notadas x, y, z, etc.) y constructores (notados c, c', etc.). Los conjuntos de variables y constructores se notan Var y $\mathcal{C}$ respectivamente. Se asume, además, que ambos son enumerables y disjuntos.

Los términos (M, N, etc.) y los case binders (θ, ϕ, etc.) de $\lambda\mathcal{B}_c$ se definen inductivamente como se muestra a continuación:

Términos	M, N	::=	x	(Variable)
		|	c	(Constructor)
		|	✠	(Daimon)
		|	MN	(Aplicación)
		|	$\lambda x\,.\,M$	(Abstracción)
		|	$\theta.M$	(Case)
Case binders	θ, ϕ	::=	$\{\!\| c_1 \mapsto M_1; \ldots; c_n \mapsto M_n \|\!\}$	$(c_i \neq c_j \text{ si } i \neq j)$

Los conjuntos de términos y de case binders se notarán $\Lambda\mathcal{B}_c$ y $\mathcal{B}$ respectivamente, y $\Lambda\mathcal{B}_c + \mathcal{B}$ su unión disjunta.

Usualmente, notaremos $\{\!\| c_i \mapsto M_i \|\!\}_{i=1}^{n}$ al case binder $\{\!\| c_1 \mapsto M_1; \cdots; c_n \mapsto M_n \|\!\}$.

1.2.5.1.1. Case binders

Cada case binder θ consiste en una lista finita no ordenada de pares $(c \mapsto M)$, cuyos lados izquierdos son mutuamente distintos. Decimos que un constructor c está *ligado* a un término M en un case binder θ si $(c \mapsto M)$ pertenece a θ. A partir de la definición de case binders, está claro que un constructor c estará ligado a no más de un término en cualquier case binder θ.

Se define una operación externa de *composición* entre dos case binders θ y ϕ, notada $\theta \circ \phi$, y definida así:

$$\theta \circ \{\!\| c_1 \mapsto M_1; \cdots; c_n \mapsto M_n \|\!\} \quad \equiv \quad \{\!\| c_1 \mapsto \theta.M_1; \cdots; c_n \mapsto \theta.M_n \|\!\}$$

donde $\phi = \{\!|\, c_1 \mapsto M_1; \cdots ; c_n \mapsto M_n \,|\!\}$.

Tal operación, si bien no es asociativa, sólo tiene sentido en presencia de la regla de reducción CaseCase, que describiremos más abajo.

1.2.5.1.2. Variables libres y sustituciones

Las nociones de ocurrencias libres y ligadas de variables se definen de la manera esperada. El conjunto de variables libres de un término M (respectivamente, un case binder θ) se nota $\mathrm{FV}(M)$ (respectivamente, $\mathrm{FV}(\theta)$) y su definición se muestra en la Figura 1.1. Se observa cómo ésta extiende de manera esperada la definición usual de variables libres.

Al igual que en el λ-cálculo clásico, los términos se consideran según α-equivalencia

(FV-Var)	$\mathrm{FV}(x)$	$=$	$\{x\}$	$(x \in \mathrm{Var})$
(FV-Cons)	$\mathrm{FV}(c)$	$=$	$\emptyset$	$(c \in \mathcal{C})$
(FV-App)	$\mathrm{FV}(M_1\, M_2)$	$=$	$\mathrm{FV}(M_1) \cup \mathrm{FV}(M_2)$	
(FV-Abs)	$\mathrm{FV}(\lambda x.M_1)$	$=$	$\mathrm{FV}(M_1) \setminus \{x\}$	
(FV-Case)	$\mathrm{FV}(\{\!|\, c_i \mapsto M_i \,|\!\}_{i=1}^{n}.N)$	$=$	$\mathrm{FV}(N) \cup \bigcup_{i=1}^{n} \mathrm{FV}(M_i)$	

Figura 1.1: Variables libres para términos de $\lambda\mathcal{B}_c$

(i.e., renombre de variables ligadas). Los nombres de los constructores no se ven afectados por tal renombre.

La operación externa de sustitución clásica $M[x \,/\, N]$ se extiende de la manera esperada a $\lambda\mathcal{B}_c$. Esta operación se introducirá formalmente en la sección 2.3.

1.2.5.2. Reglas de reducción

Este cálculo posee 9 reglas de reducción primitivas, tal como ilustra la Figura 1.2.

A los fines de la prueba de confluencia, es de interés considerar no sólo el sistema inducido por las 9 reglas en conjunto, sino también los subsistemas formados por cada subconjunto de dichas reglas.

Se nota $\lambda\mathcal{B}_c$ al cálculo generado por todas las reglas de la Figura 1.2, y $\mathcal{B}_\mathcal{C}$ al cálculo generado por todas las reglas excepto AppLam (o, también, β).

Notar que AppLam (o β) y LamApp (o η) son las únicas reglas que podrán aplicarse sobre λ-términos ordinarios en $\lambda\mathcal{B}_c$.

1.2.5.3. Confluencia de $\lambda\mathcal{B}_\mathcal{C}$

Los autores, en [5], prueban un resultado más general mediante la caracterización de cuáles subconjuntos de las 9 reglas de reducción inducen un subsistema de $\lambda\mathcal{B}_c$ que es confluente y cuáles no, tomando en consideración los $2^9 = 512$ subconjuntos posibles de tales reglas.

Cada una de las reglas describe la interacción entre dos construcciones sintácticas del lenguaje, que queda reflejada por el nombre de la regla: AppLam se entiende como "aplicación sobre lambda", etc. Estas reglas dan lugar a 13 pares críticos diferentes.

Los pares críticos ocurren para todos los pares de reglas de la forma FooBar y BarBaz. Si uno examinase los pares críticos, se daría cuenta de que, cada vez que fuese necesario cerrar cierto par, debe utilizarse la tercera regla FooBaz en caso de que exista. En los otros

Beta-reducción					
AppLam	(AL)	$(\lambda x\,.\,M)N$	$\to$	$M\{x := N\}$	
AppDai	(AD)	$\maltese\, N$	$\to$	$\maltese$	
Eta-reducción					
LamApp	(LA)	$\lambda x\,.\,Mx$	$\to$	M	$(x \notin \mathrm{FV}(M))$
LamDai	(LD)	$\lambda x\,.\maltese$	$\to$	$\maltese$	
Propagación de CBs					
CaseCons	(CO)	$\theta.c$	$\to$	M	$((c \mapsto M) \in \theta)$
CaseDai	(CD)	$\theta.\maltese$	$\to$	$\maltese$	
CaseApp	(CA)	$\theta.(MN)$	$\to$	$\theta.M\,N$	
CaseLam	(CL)	$\theta.(\lambda x\,.\,M)$	$\to$	$\lambda x\,.\,\theta.M$	$(x \notin \mathrm{FV}(\theta))$
Conversión de CBs					
CaseCase	(CC)	$\theta.\phi.M$	$\to$	$(\theta \circ \phi).M$	

Figura 1.2: Reglas de reducción de $\lambda\mathcal{B}_c$

casos, el par crítico se cierra únicamente a través de las reglas FooBar y BarBaz.

Esta observación sugiere la siguiente definición:

Definición 1.2.4 (Condiciones de clausura)**.** Decimos que un subconjunto de las 9 reglas de la Figura 1.2 satisface las *condiciones de clausura*, notado $s \models \mathrm{CC}$, si:

(CC1)	AppLam	$\in s$	$\wedge$	LamDai	$\in s$	$\Rightarrow$	AppDai	$\in s$
(CC2)	LamApp	$\in s$	$\wedge$	AppDai	$\in s$	$\Rightarrow$	LamDai	$\in s$
(CC3)	CaseApp	$\in s$	$\wedge$	AppLam	$\in s$	$\Rightarrow$	CaseLam	$\in s$
(CC4)	CaseApp	$\in s$	$\wedge$	AppDai	$\in s$	$\Rightarrow$	CaseDai	$\in s$
(CC5)	CaseLam	$\in s$	$\wedge$	LamApp	$\in s$	$\Rightarrow$	CaseApp	$\in s$
(CC6)	CaseLam	$\in s$	$\wedge$	LamDai	$\in s$	$\Rightarrow$	CaseDai	$\in s$

Intuitivamente, un subconjunto que satisface estas 6 condiciones de clausura define un sistema en donde todos los pares críticos pueden cerrarse y, por ende, constituye un buen candidato para la confluencia. El siguiente teorema indica que esa intuición es correcta:

Teorema 1.2.2 (Church-Rosser)**.** Para cada subsistema s de $\lambda\mathcal{B}_\mathcal{C}$, las siguientes afirmaciones son equivalentes:

1. s satisface las condiciones de clausura (CC1)–(CC6);
2. s es débilmente confluente;
3. s es confluente.

Como el sistema completo $\lambda\mathcal{B}_\mathcal{C}$ satisface trivialmente todas las condiciones de clausura, se obtiene como consecuencia inmediata:

Corolario 1.2.1 (Church-Rosser)**.** $\lambda\mathcal{B}_\mathcal{C}$ es confluente.

La prueba del Teorema 1.2.2 recae en un análisis sistemático de las propiedades de conmutación sobre todos los pares (s_1, s_2) de subsistemas de $\lambda\mathcal{B}_\mathcal{C}$.

Además, vale la

Proposición 1.2.3 (SN de $\mathcal{B}_\mathcal{C}$)**.** El $\mathcal{B}_\mathcal{C}$-cálculo es SN.

En lo que sigue, no utilizaremos el daimon ni las reglas de reducción asociadas (AppDai, LamDai y CaseDai), así como tampoco la regla CaseCase, que tiene su razón de ser en la propiedad de separación débil de $\lambda\mathcal{B}_c$ [5], de la cual no nos ocuparemos en la presente tesis.

Capítulo 2

Traducción de $\lambda\mathcal{C}$ a $\lambda\mathcal{B}_\mathcal{C}$

2.1. Introducción

El propósito de este capítulo es definir y estudiar distintos aspectos de una traducción que nos permitirá convertir términos del cálculo $\lambda\mathcal{C}$ a términos del cálculo $\lambda\mathcal{B}_c$, de modo que se preserven las propiedades fundamentales.

Es de interés estudiar conversiones entre distintos formalismos pues el contar con (buenas) traducciones permite lograr distintos objetivos, en mayor o menor grado. Entre ellos, cabe mencionar:

1. la comparación de poderes expresivos, el estudio de la conveniencia de una u otra sintaxis;
2. la compilación, es decir, basarse en implementaciones de un cálculo para obtener implementaciones de otro, y
3. una mejor comprensión de los formalismos de acuerdo a las relaciones que guardan.

En este caso puntual, el cálculo de origen, $\lambda\mathcal{C}$, oculta en su regla de reducción una operación compleja de matching sobre una cantidad arbitraria de patrones. En contrapartida, $\lambda\mathcal{B}_\mathcal{C}$ sólo dispone de análisis de casos por constructores (case binders) y de abstracciones clásicas junto con reglas de propagación de case binders, que dotan al cálculo de un poder expresivo que se ve inhibido en $\lambda\mathcal{C}$ por su restrictivo mecanismo de pattern matching. Esta notable diferencia en la naturaleza de ambos formalismos motiva la búsqueda de posibles traducciones entre ambos.

A modo ilustrativo, consideremos los siguientes términos de $\lambda\mathcal{C}$, asumiendo que contamos con los constructores que allí aparecen y con la interpretación usual de los mismos:

$$\begin{array}{lcccl} \text{pred} & = & \lambda\mathbf{0}.\mathbf{0} & | & \lambda\mathbf{s}(x).x \\ \text{empty} & = & \lambda\mathbf{nil}.\mathbf{true} & | & \lambda\mathbf{cons}(x,y).\mathbf{false} \end{array}$$

Estos términos representan, respectivamente, funciones que calculan el predecesor de un número natural cualquiera (tomando a 0 como su propio predecesor) y que determinan si una lista es vacía. Es de esperar que una traducción adecuada mapee dichos términos a, por ejemplo,

$$\begin{array}{lcl} \text{pred}_1 & = & \lambda v.\{\!|\, \mathbf{0} \mapsto \mathbf{0}\,;\, \mathbf{s} \mapsto \lambda x.x \,|\!\}.v \\ \text{empty}_1 & = & \lambda v.\{\!|\, \mathbf{nil} \mapsto \mathbf{true}\,;\, \mathbf{cons} \mapsto \lambda xy.\mathbf{false} \,|\!\}.v \end{array}$$

Claramente, tanto pred como empty junto con sus traducciones se comportan de la manera esperada al ser aplicados a naturales o a listas respectivamente. Ahora bien, veamos

qué sucede si el argumento no es simplemente una constante sino una *función*. En tales escenarios, el matching en $\lambda\mathcal{C}$ fallará (dado que una abstracción múltiple no es instancia de ningún patrón) y el término quedará bloqueado, mientras que las reglas de propagación de case binders de $\lambda\mathcal{B}_c$ permitirán reducciones adicionales que operen sobre la función tomada como argumento. Por ejemplo, sea $F = \lambda x.\mathbf{cons}\,\mathbf{0}\,\mathbf{nil} \in \Lambda\mathcal{B}_c$ la función que devuelve constantemente la lista que contiene únicamente a cero. Entonces,

$$
\begin{array}{lcl}
\mathrm{empty}_1\, F & = & (\lambda v.\{\!|\, \mathbf{nil} \mapsto \mathbf{true}\,;\, \mathbf{cons} \mapsto \lambda xy.\mathbf{false}\,|\!\}.v)\,(\lambda x.\mathbf{cons}\,\mathbf{0}\,\mathbf{nil}) \\
 & \overset{\textsc{AppLam}}{\longrightarrow} & \{\!|\, \mathbf{nil} \mapsto \mathbf{true}\,;\, \mathbf{cons} \mapsto \lambda xy.\mathbf{false}\,|\!\}.(\lambda x.\mathbf{cons}\,\mathbf{0}\,\mathbf{nil}) \\
 & \overset{\textsc{CaseLam}}{\longrightarrow} & \lambda x.\{\!|\, \mathbf{nil} \mapsto \mathbf{true}\,;\, \mathbf{cons} \mapsto \lambda xy.\mathbf{false}\,|\!\}.(\mathbf{cons}\,\mathbf{0}\,\mathbf{nil}) \\
 & \overset{\textsc{CaseApp}}{\longrightarrow} & \lambda x.\{\!|\, \mathbf{nil} \mapsto \mathbf{true}\,;\, \mathbf{cons} \mapsto \lambda xy.\mathbf{false}\,|\!\}.(\mathbf{cons}\,\mathbf{0})\,\mathbf{nil} \\
 & \overset{\textsc{CaseApp}}{\longrightarrow} & \lambda x.\{\!|\, \mathbf{nil} \mapsto \mathbf{true}\,;\, \mathbf{cons} \mapsto \lambda xy.\mathbf{false}\,|\!\}.\mathbf{cons}\,\mathbf{0}\,\mathbf{nil} \\
 & \overset{\textsc{CaseCons}}{\longrightarrow} & \lambda x.(\lambda xy.\mathbf{false})\,\mathbf{0}\,\mathbf{nil} \\
 & \overset{\textsc{AppLam}}{\longrightarrow} & \lambda x.(\lambda y.\mathbf{false})\,\mathbf{nil} \\
 & \overset{\textsc{AppLam}}{\longrightarrow} & \lambda x.\mathbf{false}
\end{array}
$$

Como puede verse, la evaluación en $\lambda\mathcal{B}_c$ arroja la función constante **false** si el argumento de empty_1 es una función que devuelve constantemente una lista no vacía. La situación es similar para el caso de pred_1: de tomar como argumento, por ejemplo, la función constante $\mathbf{s}(\mathbf{0})$, entonces devolverá la función constante $\mathbf{0}$.

Se observa, pues, que los términos traducidos poseen, potencialmente, una mayor prestación en relación a los términos originales de $\lambda\mathcal{C}$. Es esta peculiaridad lo que nos impulsó a formular la traducción que presentaremos.

El resto del capítulo se estructura de la siguiente manera: en la sección 2.2 discutiremos importantes características de la traducción propuesta, tomando como referencia el impacto en la misma causado por la gran diferencia entre ambos cálculos. Luego, se darán diversas definiciones que serán necesarias durante el desarrollo del capítulo (sección 2.3). A partir de ellas, estaremos en condiciones de presentar la función de traducción, lo cual se hará en la sección 2.4. A continuación, en la sección 2.5, probaremos que la traducción se comporta de la manera esperada, esto es, que permite simular tanto la reducción β-multiple-pattern y, más aún, en una cantidad lineal de pasos respecto del patrón instanciado en el redex, como η-reducción. En la sección 2.6 estudiaremos la complejidad espacial de la traducción a partir del tamaño de los términos traducidos en función del tamaño de los términos originales, y demostraremos que el peor caso es lineal. También presentaremos pruebas empíricas que no sólo sustentan esto sino que, además, muestran una constante lineal menor para un conjunto aleatorio de términos. En la sección 2.7, demostraremos los lemas auxiliares utilizados en las secciones previas y, finalmente, discutiremos algunos aspectos adicionales del desarrollo en la sección 2.8.

2.2. Características y alcance de la traducción

La traducción será definida a través de una función $\Psi : \mathfrak{M} \subset \Lambda\mathcal{C} \to \Lambda\mathcal{B}_c$, donde $\mathfrak{M}$ es un conjunto a definir. Se asume que trabajamos con el mismo conjunto de variables Var y el mismo conjunto de constructores $\mathcal{C}$ en ambos cálculos. Además, cada $c \in \mathcal{C}$ tiene asociada una *aridad*, entero no negativo, que notaremos con δ.

2.2.1. Restricción sobre el dominio

La función Ψ que proponemos parte de la base de que el término que recibe como argumento satisface una serie de condiciones. Esto no se debe a una deficiencia en la traducción, sino a la diferente naturaleza de ambos cálculos. Informalmente, el dominio de Ψ es el conjunto $\mathfrak{M}$ de términos M que satisfacen simultáneamente:

- Cada patrón que aparezca en M debe ser lineal.
- Para cada aparición de cada constructor c en M se debe respetar su aridad, $\delta(c)$.
- Cada construcción de abstracciones múltiples en M debe ser tal que los patrones que en ella aparecen sean no unificables de a pares y, más aún, debe existir una manera de poder *distinguir* cada patrón de los restantes según los constructores que en él aparecen.

Por ejemplo, el término

$$\lambda f(x, g(y, a, a)).x \mid \lambda f(a, g(b, y, b)).y \mid \lambda f(b, g(c, c, y)).y$$

no podrá ser traducido pues viola la tercera cláusula: el primer argumento de los patrones no permite distinguirlos ya que en el primero de ellos aparece una variable, y en el segundo, si bien los subpatrones son no unificables de a pares, ninguna posición permite tomar una decisión acerca de cómo distinguirlos: en las tres aparecen variables.

Por otro lado, el término

$$\lambda f(b, g(a, a, x)).x \mid \lambda f(b, g(b, a, y)).y \mid \lambda f(a, g(a, a, x)).x$$

sí admite traducción: el primer argumento de f permite distinguir al tercer patrón de los dos restantes, mientras que, para distinguir entre ellos, bastará con observar cuál es el primer argumento de g.

La idea detrás de esta condición es la de generar un case binder dentro del cual se diferencie en primera instancia cada patrón de los restantes, para luego continuar con cada uno por separado.

Notar que, en todos los casos, los cuerpos de las abstracciones son irrelevantes: no determinan la posibilidad de traducción, mientras que los patrones son quienes la determinan. Esto hace comprobar que este tipo de traducciones constituye una prueba importante en cuanto al uso de patrones desde distintos formalismos.

2.2.2. Alcance

Antes de dar la definición de la función de traducción, señalaremos brevemente el alcance de la misma, tomando como referencia lo discutido en la sección anterior. A priori, podría parecer que la cláusula de distinguibilidad se trata de una condición restrictiva, y uno podría preguntarse si no sería suficiente sólo con pedir no unificabilidad de pares (entre patrones en la misma construcción de abstracciones múltiples), tal como se hace en [58]. El problema fundamental radica, como ya se ha dicho, en la diferente naturaleza de ambos cálculos. La regla de reducción β-multiple-pattern de $\lambda\mathcal{C}$ oculta un mecanismo complejo en el cual se determina, mediante sustituciones, qué patrón el argumento instancia. Desde el punto de vista formal, esta acción es atómica y simultánea. No obstante, las reglas de $\lambda\mathcal{B}_c$ no

permiten tales reducciones: en él, la determinación del patrón instanciado necesariamente deberá realizarse en forma secuencial, a través de case binders.

En el caso más general de la no unificabilidad de a pares, esta notable diferencia puede traer inconvenientes al intentar mapear los términos a $\lambda\mathcal{B}_c$. Consideremos, a modo ilustrativo, el siguiente término:

$$\lambda f(x, a, a).x \mid \lambda f(b, x, b).x \mid \lambda f(c, c, x).x$$

Claramente, los tres patrones en cuestión no unifican de a pares. Sin embargo, no está del todo claro cómo llevar tal término a $\lambda\mathcal{B}_c$: la presencia de *variables* en cada argumento de f imposibilita predicar sobre alguno de ellos en un case binder (i.e., dado un M que instancie alguno de estos patrones, en el argumento correspondiente a x uno puede esperar *cualquier* término posible, lo cual es un obstáculo serio en $\lambda\mathcal{B}_c$ para determinar, en forma sistemática, cuál caso se debería tener en cuenta y cuáles descartar).

De cualquier manera, encontramos que la traducción que vamos a dar tiene utilidad e interés en sí misma ya que el conjunto de términos a los que puede ser aplicada es más que suficiente en la programación habitual, lo cual se sustenta con el siguiente ejemplo general. Supongamos que tenemos un conjunto $\{C_1, \cdots, C_n, R_1, \cdots, R_m\} \subseteq \mathcal{C}$ de constructores, donde $\delta(C_i) = k_i$ y $\delta(R_j) = l_j + t_j$ $(1 \leq i \leq n,\ 1 \leq j \leq m)$ y, a partir de él, definimos un conjunto de términos $\mathcal{T} \subset \Lambda\mathcal{C}$ en forma inductiva de la siguiente manera:

- $C_i(M_1, \cdots, M_{k_i}) \in \mathcal{T}$, para cada $i = 1, \cdots, n$, y $M_1, \cdots, M_{k_i} \in \Lambda\mathcal{C}$
- $R_j(M_1, \cdots, M_{l_j}, T_1, \cdots, T_{t_j}) \in \mathcal{T}$, para cada $j = 1, \cdots, m$, $M_1, \cdots, M_{l_j} \in \Lambda\mathcal{C}$ y $T_1, \cdots, T_{t_j} \in \mathcal{T}$

Tal conjunto $\mathcal{T}$ tiene sintácticamente la misma clase de estructura que un tipo algebraico definido mediante el conjunto de patrones de $\lambda\mathcal{C}$. Así, cualquier función recursiva definida por pattern matching sobre las distintas formas de construir un término de $\mathcal{T}$ admitirá traducción. El motivo es sumamente sencillo: en una construcción de abstracciones múltiples que defina tal función, cada patrón comenzará con un constructor diferente según el caso, lo cual garantiza distinguibilidad. Notar además que, en caso de existir solapamiento parcial entre patrones (como podría ser el caso de una función recursiva sobre árboles binarios no vacíos) siempre deberá existir una manera de poder diferenciar cada caso según los constructores que se utilizan.

Por ejemplo, sea $\mathcal{T} = \textsc{BinTree}$ el conjunto de árboles binarios definido a través de las reglas mostradas a continuación, considerando los constructores Nil y Bin tales que $\delta(\text{Nil}) = 0$ y $\delta(\text{Bin}) = 3$:

- $\text{Nil} \in \textsc{BinTree}$
- $\text{Bin}(M, T_1, T_2) \in \textsc{BinTree} \forall\ M \in \Lambda\mathcal{C},\ \forall\ T_1, T_2 \in \textsc{BinTree}$

Si se quisiera codificar, por ejemplo, una función que calculase el máximo elemento de un árbol binario t de naturales, se deberían plantear los siguientes casos en una abstracción múltiple:

- $\text{Bin}(x, \text{Nil}, \text{Nil})$
- $\text{Bin}(x, \text{Bin}(i, r, d), \text{Nil})$

- $\mathrm{Bin}(x, \mathrm{Nil}, \mathrm{Bin}(i, r, d))$
- $\mathrm{Bin}(x, \mathrm{Bin}(l, q, s), \mathrm{Bin}(i, r, d))$

Se puede apreciar cómo los cuatro patrones involucrados son mutuamente distinguibles.

2.2.3. Convención de variables

En lo que sigue, asumiremos que cada término M respeta la convención de variables de Barendregt. Esto se acepta porque, dado M, siempre se podrá encontrar otro término N α-equivalente a M que sí la respete, y luego continuar trabajando con éste.

También vamos a asumir que las variables de los términos, antes de ser traducidos, proceden de un conjunto $\mathcal{V}$. Durante el proceso de traducción, se introducirán distintos tipos (disjuntos) de variables. Como muestra la Definición 2.3.2, la unión de estos conjuntos determina el conjunto Var.

2.3. Definiciones previas

En esta sección definiremos conceptos que serán utilizados a lo largo del capítulo, y que además servirán para formalizar lo expresado anteriormente.

Definición 2.3.1. El *conjunto de patrones* $\mathcal{P}$ se define inductivamente a través de los siguientes axiomas:

1. $x \in \mathrm{Var} \Rightarrow x \in \mathcal{P}$
2. $c \in \mathcal{C} \wedge \delta(c) = n \wedge t_1, \cdots, t_n \in \mathcal{P} \Rightarrow c(t_1, \cdots, t_n) \in \mathcal{P}$

Definición 2.3.2. Sea $p \in \mathcal{P}$.

- El conjunto $\mathcal{V}$ representa el conjunto infinito numerable de *variables iniciales*, esto es, aquellas variables mediante las cuales se construirán los términos originales de $\lambda\mathcal{C}$ antes de ser traducidos.
- Cada símbolo de la forma $\square_p$ es una variable tal que $\square_p \notin \mathcal{V}$. Notaremos $\mathfrak{B}$ al conjunto de estas variables.
- Dados infinitos y numerables símbolos $u, v, \cdots$ que no aparezcan en los conjuntos $\mathcal{V}$ y $\mathfrak{B}$, y dada $s \in \mathbb{N}^*$, cada símbolo de la forma α_s representará una variable, donde α denota cualquiera de los símbolos mencionados $(u, v, \cdots)$. Llamaremos $\mathfrak{V}$ al conjunto de ellas.
- El conjunto Var de variables se define como la unión disjunta de estos tres conjuntos de variables:

$$\mathrm{Var} = \mathcal{V} \cup \mathfrak{V} \cup \mathfrak{B}$$

Definición 2.3.3. Sea $p \in \mathcal{P} \setminus \mathrm{Var}$, $p = c(t_1, \cdots, t_n)$. El *constructor de* p se define como $\boldsymbol{c}\,(c(t_1, \cdots, t_n)) = c$

Definición 2.3.4. Sea $p \in \mathcal{P}$. El *conjunto de posiciones de* p, $\mathrm{Pos}(p) \subset \mathbb{N}^*$, se define así:

$$\begin{array}{lcl} \mathrm{Pos}(x) & = & \{\epsilon\} \\ \mathrm{Pos}\left(c(t_1, \cdots, t_n)\right) & = & \{\epsilon\} \cup \bigcup\limits_{1 \leq i \leq n} \{\, i \cdot s \,/\, s \in \mathrm{Pos}(t_i) \,\} \end{array}$$

Definición 2.3.5. Sea $p \in \mathcal{P}$, y sea $s \in \mathrm{Pos}(p)$. La *proyección* $p|_s$ denota el subpatrón de p en la posición s:

$$\begin{array}{lcl} p|_\epsilon & = & p \\ c(t_1, \cdots, t_n)|_{i \cdot s'} & = & t_i|_{s'} \end{array}$$

Definición 2.3.6. Sean $M, N \in \Lambda\mathcal{C}$. Diremos que $M \subseteq N$ si M es *subtérmino* de N.

Esta noción de subtérmino se formaliza de la manera esperada, como extensión a la noción clásica de subtérmino. Lo mismo ocurre para $\lambda\mathcal{B}_c$.

Definición 2.3.7. Sea $M \in \Lambda\mathcal{C}$. El *conjunto de patrones* de M, $\boldsymbol{P}(M)$, se define como sigue:

$$\begin{array}{lcl} \boldsymbol{P}(M) & = & \{\, p \in \mathcal{P} \setminus \mathrm{Var} \,/\, \exists\, p_1, \cdots, p_k, M_1, \cdots, M_k : (\lambda p_1.M_1 \mid \cdots \mid \lambda p_k.M_k) \subseteq M \wedge \\ & & p = p_j \text{ para algún } j = 1, \cdots, k \,\} \end{array}$$

Definición 2.3.8. Sea $P \subseteq \mathcal{P} \setminus \mathrm{Var}$ y $s \in \bigcap\limits_{p \in P} \mathrm{Pos}(p)$ tal que $p|_s \notin \mathrm{Var}$ para cualquier $p \in P$. La relación binaria $\mathcal{R}_s$ sobre el conjunto P se define como:

$$p_1 \,\mathcal{R}_s\, p_2 \overset{\text{def}}{\Leftrightarrow} \begin{cases} s = \epsilon \,\wedge\, \boldsymbol{c}(p_1) = \boldsymbol{c}(p_2) \\ \qquad\qquad \vee \\ s = s' \cdot i \,\wedge\, \boldsymbol{c}(p_1|_s) = \boldsymbol{c}(p_2|_s) \,\wedge\, p_1 \,\mathcal{R}_{s'}\, p_2 \end{cases}$$

Puede demostrarse en forma inmediata que, para cada $s \in \bigcap\limits_{p \in P} \mathrm{Pos}(p)$, $\mathcal{R}_s$ es una relación de equivalencia sobre P.

Usaremos la notación $S/\mathcal{R}$ para representar el conjunto S cocientado por la relación de equivalencia $\mathcal{R}$.

Definición 2.3.9. El *orden de diccionario* $\lhd$ en $\mathbb{N}^*$ se define de la siguiente manera, siendo $s, t \in \mathbb{N}^*$:

$$s \lhd t \overset{\text{def}}{\Leftrightarrow} \begin{cases} t = s \cdot t' \\ \qquad \vee \\ s = s_1 \cdots s_n \,\wedge\, t = s_1 \cdots s_j t_1 \cdots t_m \,\wedge\, s_{j+1} < t_1 \,(0 \leq j < n) \end{cases}$$

Aquí, hemos agregado reflexividad a la definición tradicional del orden de diccionario. Por conveniencia, lo utilizaremos de esta manera.

En lo que sigue, y a lo largo de todo el documento, la notación $|S|$ hace referencia al cardinal de S, siendo S un conjunto.

Definición 2.3.10. Sea $P \subseteq \mathcal{P} \setminus \mathrm{Var}$ y sea $S \subseteq \mathbb{N}^*$. El conjunto $\Delta_S(P)$ contiene aquellas posiciones que no están en S y que son viables para distinguir los patrones de P, y se define así:

$$\begin{array}{lcl} \Delta_S(\{p\}) & = & \mathrm{Pos}(p) \setminus S \\ \Delta_S(P) & = & \left\{ s \in \bigcap_{p \in P} \mathrm{Pos}(p) \backslash S / \forall p \in P : p|_s \notin \mathrm{Var} \wedge \forall P' \in P / \mathcal{R}_s : \left(|P'| = 1 \vee |\Delta_{S \cup \{s\}}(P')| > 0 \right) \right\} \\ & & \text{si } |P| > 1 \end{array}$$

La intuición detrás de Δ es la de calcular posiciones s en común a todos los patrones de P tales que, además de no estar presentes en S, permitan eventualmente distinguir cada patrón. Para ello, se agrupan los patrones según la relación $\mathcal{R}_s$ (i.e., comparando los constructores en cada posición prefijo de s), dando lugar a una o más clases de equivalencia. Cada una de ellas debe ser unitaria (en cuyo caso el objetivo queda satisfecho) o bien la invocación recursiva de Δ sobre ella, agregando s a las posiciones ya tenidas previamente en cuenta, debe arrojar por lo menos una posición.

Por ejemplo, si $P = \{f(a, x, g(y, b)), f(b, a, g(b, a)), f(b, c, g(a, a))\}$, entonces $\Delta_{\{\epsilon\}}(P) = \{1, 3, 32\}$: esto dice que, una vez procesado el constructor inicial (i.e., el que aparece en la raíz de los patrones), cada una de estas posiciones puede tomarse como punto de partida dentro de la generación de un case binder para ir distinguiendo los patrones de P. Observar que cada una de ellas es tal que ningún patrón posee una variable allí: como ya se ha dicho, son los constructores los que nos permitirán tomar las decisiones a la hora de construir un término de $\lambda\mathcal{B}_c$.

Para asegurar que la traducción esté bien definida, vamos a tomar en todos los casos el mínimo de dicho conjunto según el orden de diccionario sobre $\mathbb{N}^*$ de la Definición 2.3.9.

Definición 2.3.11. Sea $M \in \Lambda\mathcal{B}_c$. La *cantidad de apariciones* de la variable x en el término M se nota con $|M|_x$ y se define por inducción en M según las ecuaciones de la Figura 2.1.

(#-Var)	$\lvert y\rvert_x$	$=$	$\begin{cases} 0 & \text{si } y \neq x \\ 1 & \text{en otro caso} \end{cases}$	$(y \in \mathrm{Var})$
(#-Cons)	$\lvert c\rvert_x$	$=$	0	$(c \in \mathcal{C})$
(#-App)	$\lvert M_1\, M_2\rvert_x$	$=$	$\lvert M_1\rvert_x + \lvert M_2\rvert_x$	
(#-Abs)	$\lvert \lambda y.M_1\rvert_x$	$=$	$\begin{cases} \lvert M_1\rvert_x & \text{si } y \neq x \\ 0 & \text{en otro caso} \end{cases}$	
(#-Case)	$\lvert \{\!\lvert\, c_i \mapsto M_i \,\rvert\!\}_{i=1}^n.N\rvert_x$	$=$	$\lvert N\rvert_x + \sum_{i=1}^{n} \lvert M_i\rvert_x$	

Figura 2.1: Cantidad de apariciones de la variable x en el término M

Definición 2.3.12. Sea $M \in \Lambda\mathcal{B}_c$. El *tamaño de* M, $|M|$, se define inductivamente como muestra la Figura 2.2[1].

De manera análoga, si $M \in \Lambda\mathcal{C}$, definimos el *tamaño de* M como indica la Figura 2.3.

Por simplicidad, usaremos la misma notación para representar ambas funciones, quedando claro la apropiada según el contexto en el que aparece.

Definición 2.3.13.

[1] Esta definición de tamaño cuenta la cantidad de nodos del árbol sintáctico asociado al término.

(S-Var-Bc)	$\lvert x\rvert$	$=$	1	$(x \in \mathrm{Var})$
(S-Cons-Bc)	$\lvert c\rvert$	$=$	1	$(c \in \mathcal{C})$
(S-App-Bc)	$\lvert M_1\, M_2\rvert$	$=$	$1 + \lvert M_1\rvert + \lvert M_2\rvert$	
(S-Abs-Bc)	$\lvert\lambda x.M_1\rvert$	$=$	$2 + \lvert M_1\rvert$	
(S-Case-Bc)	$\lvert\{\!\lvert\, c_i \mapsto M_i \,\rvert\!\}_{i=1}^{n}.N\rvert$	$=$	$2 + 2n + \lvert N\rvert + \sum_{i=1}^{n} \lvert M_i\rvert$	

Figura 2.2: Tamaño de un término de $\lambda\mathcal{B}_c$

(S-Var-C)	$\lvert x\rvert$	$=$	1	$(x \in \mathrm{Var})$
(S-Pat-C)	$\lvert c(M_1, \cdots, M_n)\rvert$	$=$	$1 + \sum_{i=1}^{n} \lvert M_i\rvert$	$(c \in \mathcal{C})$
(S-App-C)	$\lvert M_1\, M_2\rvert$	$=$	$1 + \lvert M_1\rvert + \lvert M_2\rvert$	
(S-Abs-C)	$\lvert(\lambda p_1.M_1 \mid \cdots \mid \lambda p_k.M_k)\rvert$	$=$	$1 + k + \sum_{i=1}^{k} (\lvert p_i\rvert + \lvert M_i\rvert)$	

Figura 2.3: Tamaño de un término de $\lambda\mathcal{C}$

- Una $\lambda\mathcal{B}_c$-*sustitución* es una función $\sigma : \mathrm{Var} \to \Lambda\mathcal{B}_c$ tal que $\sigma(x) \neq x$ para una cantidad finita de variables x.
- El conjunto (finito) de variables que σ no mapea a sí mismas se llama *dominio* de σ:

$$\mathrm{Dom}(\sigma) = \{\, x \in \mathrm{Var} \,/\, \sigma(x) \neq x \,\}$$

- El *rango de variables* de σ es un conjunto formado por las variables libres que aparecen en los términos a los que σ mapea las variables en su dominio:

$$\mathrm{VRan}(\sigma) = \bigcup_{x \in \mathrm{Dom}(\sigma)} \mathrm{FV}(\sigma(x))$$

- Una $\lambda\mathcal{B}_c$-sustitución σ puede *extenderse* a un mapeo $\hat{\sigma} : \Lambda\mathcal{B}_c \to \Lambda\mathcal{B}_c$ tal como muestra la Figura 2.4.

(Sub-Var-Bc)	$\hat{\sigma}(x)$	$=$	$\sigma(x)$	$(x \in \mathrm{Var})$
(Sub-Cons-Bc)	$\hat{\sigma}(c)$	$=$	c	$(c \in \mathcal{C})$
(Sub-App-Bc)	$\hat{\sigma}(M_1\, M_2)$	$=$	$\hat{\sigma}(M_1)\, \hat{\sigma}(M_2)$	
(Sub-Abs-Bc)	$\hat{\sigma}(\lambda x.M_1)$	$=$	$\lambda x.\hat{\sigma}(M_1)$	
(Sub-Case-Bc)	$\hat{\sigma}(\{\!\lvert\, c_i \mapsto M_i \,\rvert\!\}_{i=1}^{n}.N)$	$=$	$\{\!\lvert\, c_i \mapsto \hat{\sigma}(M_i) \,\rvert\!\}_{i=1}^{n}.\hat{\sigma}(N)$	

Figura 2.4: Sustitución sobre términos de $\lambda\mathcal{B}_c$

(Sub-Var-C)	$\hat{\sigma}(x)$	$=$	$\sigma(x)$	$(x \in \text{Var})$
(Sub-Pat-C)	$\hat{\sigma}(c(M_1, \cdots, M_n))$	$=$	$c\,(\hat{\sigma}(M_1), \cdots, \hat{\sigma}(M_n))$	$(c \in \mathcal{C})$
(Sub-App-C)	$\hat{\sigma}(M_1\, M_2)$	$=$	$\hat{\sigma}(M_1)\,\hat{\sigma}(M_2)$	
(Sub-Abs-C)	$\hat{\sigma}(\lambda p_1.M_1 \mid \cdots \mid \lambda p_k.M_k)$	$=$	$\lambda p_1.\hat{\sigma}(M_1) \mid \cdots \mid \lambda p_k.\hat{\sigma}(M_k)$	

Figura 2.5: Sustitución sobre términos de $\lambda\mathcal{C}$

Las mismas definiciones pueden darse en forma análoga con respecto a $\lambda\mathcal{C}$. La Figura 2.5 muestra el caso del mapeo extendido.

Nombraremos simplemente *sustituciones* a las $\lambda\mathcal{B}_c$-sustituciones. Usualmente, utilizaremos la notación M^σ como sinónimo de $\sigma(M)$. Además, en algunos casos escribiremos una sustitución σ con dominio $\text{Dom}(\sigma) = \{\, x_1, \cdots, x_n \,\}$ como $\sigma = \{\, x_1 \mapsto \sigma(x_1), \cdots, x_n \mapsto \sigma(x_n) \,\}$. Por simplicidad, llamaremos de igual manera a una sustitución σ y a su extensión $\hat{\sigma}$.

La sustitución habitual de $\lambda\mathcal{B}_c$, que notamos $M[x \,/\, N]$, la tomaremos como sinónimo de una $\lambda\mathcal{B}_c$-sustitución $\tau = \{x \mapsto N\}$ aplicada a M, M^τ. El contar con α-conversión y con la convención de variables permite trabajar de esta forma al evitar capturas.

2.4. La función de traducción Ψ

Con las definiciones presentadas en la sección 2.3, ya estamos en condiciones de formalizar la función $\Psi : \mathfrak{M} \to \Lambda\mathcal{B}_c$. En primer lugar, definiremos su dominio, el conjunto $\mathfrak{M}$, y haremos algunas observaciones sobre él en la siguiente subsección.

2.4.1. Dominio

La Definición 2.4.1 presenta formalmente el conjunto $\mathfrak{M}$.

Definición 2.4.1. El conjunto $\mathfrak{M} \subset \Lambda\mathcal{C}$ es el conjunto formado por los términos M que satisfacen en forma simultánea las siguientes condiciones:

i. $|p|_x \leq 1 \;\forall\, p \in \boldsymbol{P}(M) \;\forall\, x \in \text{Var}$

ii. si $c(M_1, \cdots, M_n) \subseteq M \;\vee\; c(t_1, \cdots, t_n) \in \boldsymbol{P}(M) \;\Rightarrow\; n = \delta(c)$

iii. si $\lambda p_1.M_1 \mid \cdots \mid \lambda p_k.M_k \;\subseteq M, k > 1 \;\Rightarrow\; |\Delta_\emptyset(\{p_1, \cdots, p_k\})| > 0$

Observemos que la cláusula *i* no supone nada extraño: como ya se comentó en la sección 1.2.4, la linealidad es una propiedad fundamental para garantizar confluencia en $\lambda\mathcal{C}$ [45, 58]. Sucede lo mismo respecto de la cláusula *ii*, donde se pide que la aridad de cada constructor sea respetada en el término, al igual que en [45]. No obstante esto, creemos firmemente que esta restricción podría eliminarse, sin perder por ello el resultado principal.

En cuanto a la cláusula *iii*, la misma asegura la distinguibilidad de los patrones involucrados en una misma abstracción múltiple. Como ya se discutió en la sección 2.2.2, esta restricción no quita utilidad o interés a la traducción. Por otro lado, contar con esta condición garantiza no unificabilidad de a pares, lo cual es imprescindible para obtener un cálculo confluente. Si bien no haremos una prueba rigurosa de esta implicación, informalmente puede entenderse de la siguiente manera: si, en una misma abstracción múltiple,

existieran dos patrones unificables p_i y p_j, los mismos deben poseer los mismos constructores en las mismas posiciones (sin considerar posiciones de variables). De esta manera, ambos patrones formarán parte de las mismas clases de equivalencia módulo $\mathcal{R}_s$, siendo s cualquiera de dichas posiciones. Así, no podrá ocurrir que Δ no sea vacío, dado que eventualmente las posiciones candidatas (i.e., aquéllas no presentes en S que no referencien variables) se agotarán y tanto p_i como p_j formarán parte del conjunto de patrones tomado como argumento por Δ.

Por otro lado, nótese que el subcálculo inducido por $\mathfrak{M}$ es cerrado por sustituciones y por reducciones. Las cláusulas previas sólo hablan sobre patrones y sobre la aridad de los constructores, cosas que no sufren modificaciones al sustituir o reducir. Esto legitimiza la utilización de este subcálculo; no es posible salirse de él una vez que se entra.

Ahora pasaremos a definir Ψ de manera inductiva de acuerdo a la estructura de su argumento. Las ecuaciones correspondientes son presentadas en la Figura 2.6.

(Ψ-Var)	$\Psi(x)$	$=$	x	$(x \in \mathrm{Var})$
(Ψ-Pat)	$\Psi(c(M_1, \cdots, M_n))$	$=$	$c\ \Psi(M_1) \cdots \Psi(M_n)$	$(c \in \mathcal{C})$
(Ψ-App)	$\Psi(M_1\ M_2)$	$=$	$\Psi(M_1)\ \Psi(M_2)$	
(Ψ-Abs)	$\Psi(\lambda p_1.M_1 \mid \cdots \mid \lambda p_k.M_k)$	$=$	$\lambda v.\Phi_v\ (\{p_1, \cdots, p_k\}, \emptyset)^\rho$	

Figura 2.6: La función Ψ

En primer lugar, obsérvese que la ecuación Ψ-Var está definida para *cualquier* variable, mientras que, anteriormente, hemos asumido que los términos de $\lambda\mathcal{C}$ toman sus variables sólo de $\mathcal{V}$ antes de ser traducidos. No obstante, no son éstas nociones incompatibles.

Con respecto a la ecuación Ψ-Abs, es preciso remarcar lo siguiente:

- $v \in \mathfrak{V}$ es una variable fresca.
- ρ es una sustitución que opera sobre las variables $\Box_{p_j}$:

$$\rho(x) = \begin{cases} \Psi(M_j^{\omega_j}) & \text{si } x = \Box_{p_j},\ 1 \leq j \leq k \\ x & \text{en otro caso} \end{cases}$$

- Cada ω_j es una sustitución que (sólo) renombra variables:

$$\omega_j(x) = \begin{cases} v_s & \text{si } x \in \mathrm{FV}(p_j) \\ x & \text{en otro caso} \end{cases}$$

  $s \in \mathbb{N}^*$ es tal que $p_j|_s = x$ (i.e., es la posición de la variable x en el patrón p_j).

Observar que, de ser algún p_i variable, entonces $k = 1$ como consecuencia de la cláusula 2.4.1*iii*.

El propósito de la variable fresca v es evitar posibles conflictos de variables ligadas en las sucesivas traducciones de cada subtérmino. En la traducción de abstracciones múltiples, a cargo de la función Φ, cada variable ligada será de la forma v_s. De esta manera, si una abstracción múltiple apareciera en forma anidada, dentro de otra, la más interna se traducirá tomando como referencia una variable $v' \neq v$, posibilitando así diferenciar cada conjunto de variables.

En cuanto a la función Φ, ésta se encarga de generar el case binder apropiado para simular la abstracción múltiple en cuestión, nombrando a cada variable utilizada en los

pasos intermedios según una variable de referencia v. Esencialmente, procede separando cada subconjunto de patrones según el constructor que aparezca en la mínima posición del conjunto Δ (Definición 2.3.10), hasta el punto en el cual cada patrón esté totalmente diferenciado de los demás, momento en el cual se continúa con su procesamiento de manera secuencial. En cada llamada recursiva, el conjunto de posiciones que se pasa como argumento se va aumentando con la posición analizada en la invocación previa, asegurándose así no repetir información en la estructura generada. La recursión finaliza cuando se agotan todas las posiciones de un patrón p dado: en este punto, se retorna una variable $\square_p$, que luego será sustituida a través de ρ por la traducción del cuerpo de la abstracción asociada a p.

El dominio de Φ es el conjunto de pares (P, S) tales que P consiste en un único patrón o bien $\Delta_S(P)$ no es vacío. En términos más formales,

$$\mathrm{Dom}(\Phi) = \{(P, S) \in \mathfrak{P}(\mathcal{P}) \times \mathfrak{P}(\mathbb{N}^*) \,/\, |P| = 1 \vee \Delta_S(P) \neq \emptyset\}$$

De esta manera, Φ queda definida mediante las ecuaciones de la Figura 2.7.

$$\begin{array}{lrcl}
(\Phi\text{-}1) & \Phi_v\left(\{p\}, S\right) & = & \begin{cases} \square_p & \text{si } \mathrm{Pos}(p) \subseteq S \\ \Phi_v\left(\{p\}, S \cup \{s\}\right) & \text{si } p|_s \in \mathcal{V} \wedge \mathrm{Pos}(p) \not\subseteq S \\ \{\!|\, \boldsymbol{c}(p|_s) \mapsto \lambda v_{s\cdot 1} \cdots v_{s\cdot n}. & \\ \Phi_v\left(\{p\}, S \cup \{s\}\right) \,|\!\}.v_s & \text{en otro caso} \end{cases} \\
 & & & \text{donde } n = \delta(\boldsymbol{c}(p|_s)) \text{ y } s = \min_{\lhd}(\Delta_S(\{p\})) \\
(\Phi\text{-}2) & \Phi_v(P, S) & = & \{\!|\, c_1 \mapsto T_1; \cdots; c_m \mapsto T_m \,|\!\}.v_s \\
 & & & \text{si } |P| > 1, \text{ con } s = \min_{\lhd}(\Delta_S(P))
\end{array}$$

Figura 2.7: La función Φ

En la ecuación Φ-2, m hace referencia a la cantidad de clases de equivalencia de P módulo $\mathcal{R}_s$. Allí, se tiene que, para cada $j = 1, \cdots, m$:

- $\forall\, p \in P_j \,:\, c_j = \boldsymbol{c}(p|_s)$, con $\{P_1, \cdots, P_m\} = P \diagup \mathcal{R}_s$
- $T_j = \lambda v_{s\cdot 1} \cdots v_{s\cdot\delta(c_j)}.\Phi_v(P_j, S \cup \{s\})$

El conjunto S contiene las posiciones de los patrones de P que ya fueron exploradas en invocaciones previas. Por este motivo, $\Delta_S(P)$ indica cuáles posiciones (en común a dichos patrones) son factibles para continuar generando la estructura del case binder, de manera de eventualmente lograr distinguir cada patrón de todos los restantes. Para que Φ esté bien definida, hemos optado por tomar siempre la mínima posición (según $\lhd$) de tal conjunto. Observar, no obstante, que cualquier otra posición podría escogerse; cada elección daría lugar a un case binder estructuralmente distinto pero, por supuesto, equivalente a los demás.

La ecuación Φ-2 define qué camino debe seguirse cuando P contiene dos o más patrones: en tal escenario, se particiona P según la relación $\mathcal{R}_s$, dando origen a m clases de equivalencia. En términos informales, esto significa que los patrones de P se agrupan de acuerdo al constructor presente en la posición s; tales constructores quedan denotados a través de $c_1, \cdots, c_m$. Luego, cada uno de los términos T_j debe tomar la cantidad de argumentos esperada por el constructor respectivo (i.e., su aridad) y, finalmente, una invocación recursiva se encarga de generar el resto del case binder para los patrones en la respectiva clase de equivalencia (i.e., aquéllos que coinciden en la posición s).

Una vez que un patrón p quede totalmente distinguido, se realizará una llamada recursiva tal que $P = \{p\}$. En tal situación, definida a través de Φ-1, sólo será necesario generar la

porción del case binder que procese la estructura de p que aún no fue tenida en cuenta. En otras palabras, Φ deberá analizar cada posición de p no presente en S. Estas posiciones son, precisamente, las indicadas por $\Delta_S(P)$[2]. Cuando la posición actual (i.e., s) sea una posición de variable en p, no habrá estructura que volcar al case binder, por lo que la evaluación continúa agregando inmediatamente s a S. Si, por otra parte, existiese un constructor en p en la posición s, se deberá reflejar tal información estructural de la manera adecuada. La recursión finalizará cuando se hayan agotado las posiciones de p sin revisar hasta el momento.

2.4.2. Algunos ejemplos

Los siguientes términos M_1 y M_2 ilustrarán el mecanismo de la función de la traducción. La metodología que utilizaremos para mostrar cómo se traducen será *bottom-up*, es decir, se traducirá en primera instancia cada subtérmino para luego construir a partir de los resultados parciales la traducción del término completo.

- $M_1 = (\lambda f(a,x,b).x \,|\, \lambda f(a,c,a).a \,|\, \lambda f(c,c,a).c) \quad f(a, g(a), b)$
- $M_2 = (\lambda g(x,y).(\lambda g(a,z).xz \,|\, \lambda g(b,z).z)\, y) \quad (g(\lambda x.x, g(a,b)))$

Para M_1 tenemos:

- x

$$\Psi(x) \overset{\Psi\text{-VAR}}{=} x$$

Esto también vale para las variables y y z.

- a

$$\Psi(a) \overset{\Psi\text{-PAT}}{=} a$$

De igual manera, esto también es cierto para cualquier constructor de aridad nula (en particular, para b y c).

- $g(a)$

$$\begin{array}{rcl} \Psi(g(a)) & \overset{\Psi\text{-PAT}}{=} & g\,\Psi(a) \\ & = & g\,a \end{array}$$

- $f(a, g(a), b)$

$$\begin{array}{rcl} \Psi(f(a,g(a),b)) & \overset{\Psi\text{-PAT}}{=} & f\,\Psi(a)\,\Psi(g(a))\,\Psi(b) \\ & = & f\,a\,(g\,a)\,b \end{array}$$

- $\Phi_v(\{f(a,x,b)\}, \{\epsilon, 1, 2, 3\})$

$$\Phi_v(\{f(a,x,b)\}, \{\epsilon, 1, 2, 3\}) \overset{\Phi\text{-1}}{=} \square_{f(a,x,b)}$$

[2]La definición de $\Delta_S(P)$ para el caso donde $|P| = 1$ garantiza este hecho.

- $\Phi_v(\{f(a,c,a)\},\{\epsilon,1,2,3\})$

$$\Phi_v(\{f(a,c,a)\},\{\epsilon,1,2,3\}) \overset{\Phi\text{-}1}{=} \Box_{f(a,c,a)}$$

- $\Phi_v(\{f(c,c,a)\},\{\epsilon,1,2,3\})$

$$\Phi_v(\{f(c,c,a)\},\{\epsilon,1,2,3\}) \overset{\Phi\text{-}1}{=} \Box_{f(c,c,a)}$$

- $\Phi_v(\{f(a,x,b)\},\{\epsilon,1,3\})$

$$\begin{array}{rcl}\Phi_v(\{f(a,x,b)\},\{\epsilon,1,3\}) & \overset{\Phi\text{-}1}{=} & \Phi_v(\{f(a,x,b)\},\{\epsilon,1,3,2\}) \\ & = & \Box_{f(a,x,b)}\end{array}$$

  - $\Delta_{\{\epsilon,1,3\}}(\{f(a,x,b)\}) = \{2\}$
  - $\min\limits_{\lhd}(\Delta_{\{\epsilon,1,3\}}(\{f(a,x,b)\})) = 2$
  - $f(a,x,b)|_2 = x \in \mathcal{V}$

- $\Phi_v(\{f(a,c,a)\},\{\epsilon,1,3\})$

$$\begin{array}{rcl}\Phi_v(\{f(a,c,a)\},\{\epsilon,1,3\}) & \overset{\Phi\text{-}1}{=} & \{\!|\, c \mapsto \Phi_v(\{f(a,c,a)\},\{\epsilon,1,3,2\})\,|\!\}.v_2 \\ & = & \{\!|\, c \mapsto \Box_{f(a,c,a)}\,|\!\}.v_2\end{array}$$

  - $\Delta_{\{\epsilon,1,3\}}(\{f(a,c,a)\}) = \{2\}$
  - $\min\limits_{\lhd}(\Delta_{\{\epsilon,1,3\}}(\{f(a,c,a)\})) = 2$
  - $f(a,c,a)|_2 = c$

- $\Phi_v(\{f(c,c,a)\},\{\epsilon,1,2\})$

$$\begin{array}{rcl}\Phi_v(\{f(c,c,a)\},\{\epsilon,1,2\}) & \overset{\Phi\text{-}1}{=} & \{\!|\, a \mapsto \Phi_v(\{f(c,c,a)\},\{\epsilon,1,2,3\})\,|\!\}.v_3 \\ & = & \{\!|\, a \mapsto \Box_{f(c,c,a)}\,|\!\}.v_3\end{array}$$

  - $\Delta_{\{\epsilon,1,2\}}(\{f(c,c,a)\}) = \{3\}$
  - $\min\limits_{\lhd}(\Delta_{\{\epsilon,1,2\}}(\{f(c,c,a)\})) = 3$
  - $f(c,c,a)|_3 = a$

- $\Phi_v(\{f(c,c,a)\},\{\epsilon,1\})$

$$\begin{array}{rcl}\Phi_v(\{f(c,c,a)\},\{\epsilon,1\}) & \overset{\Phi\text{-}1}{=} & \{\!|\, c \mapsto \Phi_v(\{f(c,c,a)\},\{\epsilon,1,2\})\,|\!\}.v_2 \\ & = & \{\!|\, c \mapsto \{\!|\, a \mapsto \Box_{f(c,c,a)}\,|\!\}.v_3\,|\!\}.v_2\end{array}$$

  - $\Delta_{\{\epsilon,1\}}(\{f(c,c,a)\}) = \{2,3\}$
  - $\min\limits_{\lhd}(\Delta_{\{\epsilon,1\}}(\{f(c,c,a)\})) = 2$
  - $f(c,c,a)|_2 = c$

- $\Phi_v(\{f(a,x,b), f(a,c,a)\}, \{\epsilon, 1\})$

$$\begin{array}{rcl}\Phi_v(\{f(a,x,b), f(a,c,a)\}, \{\epsilon,1\}) & \overset{\Phi\text{-}2}{=} & \{\!|\, b \mapsto \Phi_v(\{f(a,x,b)\}, \{\epsilon,1,3\}) \\ & & \;;\, a \mapsto \Phi_v(\{f(a,c,a)\}, \{\epsilon,1,3\}) \,|\!\}.v_3 \\ & = & \{\!|\, b \mapsto \Box_{f(a,x,b)} \,;\, a \mapsto \{\!|\, c \mapsto \Box_{f(a,c,a)} \,|\!\}.v_2 \,|\!\}.v_3\end{array}$$

  - $\Delta_{\{\epsilon,1\}}(\{f(a,x,b), f(a,c,a)\}) = \{3\}$
  - $\min\limits_{\lhd}(\Delta_{\{\epsilon,1\}}(\{f(a,x,b), f(a,c,a)\})) = 3$
  - $\{f(a,x,b), f(a,c,a)\} \diagup_{\mathcal{R}_3} = \{\{f(a,x,b)\}, \{f(a,c,a)\}\}$

- $\Phi_v(\{f(a,x,b), f(a,c,a), f(c,c,a)\}, \{\epsilon\})$

$$\begin{array}{rcl}\Phi_v(\{f(a,x,b), f(a,c,a), f(c,c,a)\}, \{\epsilon\}) & \overset{\Phi\text{-}2}{=} & \{\!|\, a \mapsto \Phi_v(\{f(a,x,b), f(a,c,a)\}, \{\epsilon,1\}) \\ & & \;;\, c \mapsto \Phi_v(\{f(c,c,a)\}, \{\epsilon,1\}) \,|\!\}.v_1 \\ & = & \{\!|\, a \mapsto \{\!|\, b \mapsto \Box_{f(a,x,b)} \\ & & \qquad\quad ;\, a \mapsto \{\!|\, c \mapsto \Box_{f(a,c,a)} \,|\!\}.v_2 \,|\!\}.v_3 \\ & & ;\, c \mapsto \{\!|\, c \mapsto \{\!|\, a \mapsto \Box_{f(c,c,a)} \,|\!\}.v_3 \,|\!\}.v_2 \,|\!\}.v_1\end{array}$$

  - $\Delta_{\{\epsilon\}}(\{f(a,x,b), f(a,c,a), f(c,c,a)\}) = \{1,3\}$
  - $\min\limits_{\lhd}(\Delta_{\{\epsilon\}}(\{f(a,x,b), f(a,c,a), f(c,c,a)\})) = 1$
  - $\{f(a,x,b), f(a,c,a), f(c,c,a)\} \diagup_{\mathcal{R}_1} = \{\{f(a,x,b), f(a,c,a)\}, \{f(c,c,a)\}\}$

- $P_1 = \Phi_v(\{f(a,x,b), f(a,c,a), f(c,c,a)\}, \emptyset)$

$$\begin{array}{rcl}P_1 & \overset{\Phi\text{-}2}{=} & \{\!|\, f \mapsto \lambda v_1 v_2 v_3.\Phi_v(\{f(a,x,b), f(a,c,a), f(c,c,a)\}, \{\epsilon\}) \,|\!\}.v \\ & = & \{\!|\, f \mapsto \lambda v_1 v_2 v_3.\{\!|\, a \mapsto \{\!|\, b \mapsto \Box_{f(a,x,b)} \\ & & \qquad\quad ;\, a \mapsto \{\!|\, c \mapsto \Box_{f(a,c,a)} \,|\!\}.v_2 \,|\!\}.v_3 \\ & & ;\, c \mapsto \{\!|\, c \mapsto \{\!|\, a \mapsto \Box_{f(c,c,a)} \,|\!\}.v_3 \,|\!\}.v_2 \,|\!\}.v_1 \,|\!\}.v\end{array}$$

  - $\Delta_{\emptyset}(\{f(a,x,b), f(a,c,a), f(c,c,a)\}) = \{\epsilon,1,3\}$
  - $\min\limits_{\lhd}(\Delta_{\emptyset}(\{f(a,x,b), f(a,c,a), f(c,c,a)\})) = \epsilon$
  - $\{f(a,x,b), f(a,c,a), f(c,c,a)\} \diagup_{\mathcal{R}_\epsilon} = \{\{f(a,x,b), f(a,c,a), f(c,c,a)\}\}$

- $N_1 = \lambda f(a,x,b).x \mid \lambda f(a,c,a).a \mid \lambda f(c,c,a).c$

$$\begin{array}{rcl}\Psi(N_1) & \overset{\Psi\text{-Abs}}{=} & \lambda v.\Phi_v(\{f(a,x,b), f(a,c,a), f(c,c,a)\}, \emptyset)^\rho \\ & = & \lambda v.(\{\!|\, f \mapsto \lambda v_1 v_2 v_3.\{\!|\, a \mapsto \{\!|\, b \mapsto \Box_{f(a,x,b)} \\ & & \qquad\quad ;\, a \mapsto \{\!|\, c \mapsto \Box_{f(a,c,a)} \,|\!\}.v_2 \,|\!\}.v_3 \\ & & ;c \mapsto \{\!|\, c \mapsto \{\!|\, a \mapsto \Box_{f(c,c,a)} \,|\!\}.v_3 \,|\!\}.v_2 \,|\!\}.v_1 \,|\!\}.v)^\rho \\ & = & \lambda v.\{\!|\, f \mapsto \lambda v_1 v_2 v_3.\{\!|\, a \mapsto \{\!|\, b \mapsto v_2 \\ & & \qquad\quad ;\, a \mapsto \{\!|\, c \mapsto a \,|\!\}.v_2 \,|\!\}.v_3 \\ & & ;c \mapsto \{\!|\, c \mapsto \{\!|\, a \mapsto c \,|\!\}.v_3 \,|\!\}.v_2 \,|\!\}.v_1 \,|\!\}.v\end{array}$$

- $\rho = \{\Box_{f(a,x,b)} \mapsto \Psi(x^{\omega_1}); \Box_{f(a,c,a)} \mapsto \Psi(a^{\omega_2}); \Box_{f(c,c,a)} \mapsto \Psi(c^{\omega_3})\}$
- $\omega_1(x) = v_2$, $\omega_1(y) = y$ para cualquier $y \neq x$
- $\omega_2(x) = x = \omega_3(x)$ para cualquier variable x

- $M_1 = (\lambda f(a,x,b).x \,|\, \lambda f(a,c,a).a \,|\, \lambda f(c,c,a).c) \quad f(a, g(a), b)$

$$\begin{array}{rcl} \Psi(M_1) & \overset{\Psi\text{-App}}{=} & \Psi(\lambda f(a,x,b).x \,|\, \lambda f(a,c,a).a \,|\, \lambda f(c,c,a).c) \quad \Psi(f(a,g(a),b)) \\ & = & (\lambda v.\{\!|\, f \mapsto \lambda v_1 v_2 v_3.\{\!|\, a \mapsto \{\!|\, b \mapsto v_2 \\ & & \qquad\qquad\qquad\qquad\qquad\qquad ;\, a \mapsto \{\!|\, c \mapsto a \,|\!\}.v_2 \,|\!\}.v_3 \\ & & \qquad\qquad\qquad\qquad\qquad ;c \mapsto \{\!|\, c \mapsto \{\!|\, a \mapsto c \,|\!\}.v_3 \,|\!\}.v_2 \,|\!\}.v_1 \,|\!\}.v) \\ & & (f\ a\ (g\ a)\ b) \end{array}$$

Para el caso de M_2, omitiremos los subtérminos simples. Además, apelando a la Proposición 2.4.1, usaremos que $\Psi(\lambda x.x) = \lambda v.v$.

- $\Phi_w(\{g(a,z)\}, \{\epsilon, 1, 2\})$

$$\Phi_w(\{g(a,z)\}, \{\epsilon,1,2\}) \overset{\Phi\text{-1}}{=} \Box_{g(a,z)}$$

- $\Phi_w(\{g(b,z)\}, \{\epsilon, 1, 2\})$

$$\Phi_w(\{g(b,z)\}, \{\epsilon,1,2\}) \overset{\Phi\text{-1}}{=} \Box_{g(b,z)}$$

- $\Phi_w(\{g(a,z)\}, \{\epsilon, 1\})$

$$\begin{array}{rcl} \Phi_w(\{g(a,z)\}, \{\epsilon,1\}) & \overset{\Phi\text{-1}}{=} & \Phi_w(\{g(a,z)\}, \{\epsilon,1,2\}) \\ & = & \Box_{g(a,z)} \end{array}$$

  - $\Delta_{\{\epsilon,1\}}(\{g(a,z)\}) = \{2\}$
  - $\min\limits_{\lhd}(\Delta_{\{\epsilon,1\}}(\{g(a,z)\})) = 2$
  - $g(b,z)|_2 = z \in \mathcal{V}$

- $\Phi_w(\{g(b,z)\}, \{\epsilon, 1\})$

$$\begin{array}{rcl} \Phi_w(\{g(b,z)\}, \{\epsilon,1\}) & \overset{\Phi\text{-1}}{=} & \Phi_w(\{g(b,z)\}, \{\epsilon,1,2\}) \\ & = & \Box_{g(b,z)} \end{array}$$

  - $\Delta_{\{\epsilon,1\}}(\{g(b,z)\}) = \{2\}$
  - $\min\limits_{\lhd}(\Delta_{\{\epsilon,1\}}(\{g(b,z)\})) = 2$
  - $g(b,z)|_2 = z \in \mathcal{V}$

- $\Phi_w(\{g(a,z), g(b,z)\}, \{\epsilon\})$

$$\begin{array}{rcl} \Phi_w(\{g(a,z),g(b,z)\}, \{\epsilon\}) & \overset{\Phi\text{-2}}{=} & \{\!|\, a \mapsto \Phi_w(\{g(a,z)\}, \{\epsilon,1\}) \\ & & \quad ;\, b \mapsto \Phi_w(\{g(b,z)\}, \{\epsilon,1\}) \,|\!\}.w_1 \\ & = & \{\!|\, a \mapsto \Box_{g(a,z)} \,;\, b \mapsto \Box_{g(b,z)} \,|\!\}.w_1 \end{array}$$

- $\Delta_{\{\epsilon\}}(\{g(a,z), g(b,z)\}) = \{1\}$
- $\min_{\lhd}(\Delta_{\{\epsilon\}}(\{g(a,z), g(b,z)\})) = 1$
- $\{g(a,z), g(b,z)\}/_{\mathcal{R}_1} = \{\{g(a,z)\}, \{g(b,z)\}\}$

▪ $P_2 = \Phi_w(\{g(a,z), g(b,z)\}, \emptyset)$

$$\begin{array}{rcl} P_2 & \overset{\Phi\text{-}2}{=} & \{\!|\, g \mapsto \lambda w_1 w_2.\Phi_w(\{g(a,z), g(b,z)\}, \{\epsilon\})\, |\!\}.w \\ & = & \{\!|\, g \mapsto \lambda w_1 w_2.\{\!|\, a \mapsto \Box_{g(a,z)}\,;\, b \mapsto \Box_{g(b,z)}\, |\!\}.w_1\, |\!\}.w \end{array}$$

- $\Delta_{\emptyset}(\{g(a,z), g(b,z)\}) = \{\epsilon, 1\}$
- $\min_{\lhd}(\Delta_{\emptyset}(\{g(a,z), g(b,z)\})) = \epsilon$
- $\{g(a,z), g(b,z)\}/_{\mathcal{R}_\epsilon} = \{\{g(a,z), g(b,z)\}\}$

▪ $N_2 = \lambda g(a,z).xz \,|\, \lambda g(b,z).z$

$$\begin{array}{rcl} \Psi(N_2) & \overset{\Psi\text{-Abs}}{=} & \lambda w.\Phi_w(\{g(a,z), g(b,z)\}, \emptyset)^\tau \\ & = & \lambda w.(\{\!|\, g \mapsto \lambda w_1 w_2.\{\!|\, a \mapsto \Box_{g(a,z)}\,;\, b \mapsto \Box_{g(b,z)}\, |\!\}.w_1\, |\!\}.w)^\tau \\ & = & \lambda w.\{\!|\, g \mapsto \lambda w_1 w_2.\{\!|\, a \mapsto x\, w_2\,;\, b \mapsto w_2\, |\!\}.w_1\, |\!\}.w \end{array}$$

- $\tau = \{\Box_{g(a,z)} \mapsto \Psi((xz)^{\mu_1}); \Box_{g(b,z)} \mapsto \Psi(z^{\mu_2})\}$
- $\mu_1(z) = \mu_2(z) = w_2$, $\mu_1(y) = \mu_2(y) = y$ para cualquier $y \neq z$

▪ $(\lambda g(a,z).xz \,|\, \lambda g(b,z).z)\, y$

$$\begin{array}{rcl} \Psi((\lambda g(a,z).xz \,|\, \lambda g(b,z).z)\, y) & \overset{\Psi\text{-App}}{=} & \Psi(\lambda g(a,z).xz \,|\, \lambda g(b,z).z)\, \Psi(y) \\ & = & (\lambda w.\{\!|\, g \mapsto \lambda w_1 w_2.\{\!|\, a \mapsto x\, w_2\,;\, b \mapsto w_2\, |\!\}.w_1\, |\!\}.w)\, y \end{array}$$

▪ $\Phi_u(\{g(x,y)\}, \{\epsilon, 1, 2\})$

$$\Phi_u(\{g(x,y)\}, \{\epsilon, 1, 2\}) \overset{\Phi\text{-}1}{=} \Box_{g(x,y)}$$

▪ $\Phi_u(\{g(x,y)\}, \{\epsilon, 1\})$

$$\begin{array}{rcl} \Phi_u(\{g(x,y)\}, \{\epsilon, 1\}) & \overset{\Phi\text{-}1}{=} & \Phi_u(\{g(x,y)\}, \{\epsilon, 1, 2\}) \\ & = & \Box_{g(x,y)} \end{array}$$

- $\Delta_{\{\epsilon,1\}}(\{g(x,y)\}) = \{2\}$
- $\min_{\lhd}(\Delta_{\{\epsilon,1\}}(\{g(x,y)\})) = 2$
- $g(x,y)|_2 = y \in \mathcal{V}$

- $\Phi_u(\{g(x,y)\},\{\epsilon\})$

$$\begin{array}{rcl}\Phi_u(\{g(x,y)\},\{\epsilon\}) & \overset{\Phi\text{-}1}{=} & \Phi_u(\{g(x,y)\},\{\epsilon,1\}) \\ & = & \Box_{g(x,y)}\end{array}$$

  - $\Delta_{\{\epsilon\}}(\{g(x,y)\}) = \{1,2\}$
  - $\min\limits_{\lhd}(\Delta_{\{\epsilon\}}(\{g(x,y)\})) = 1$
  - $g(x,y)|_1 = x \in \mathcal{V}$

- $\Phi_u(\{g(x,y)\},\emptyset)$

$$\begin{array}{rcl}\Phi_u(\{g(x,y)\},\emptyset) & \overset{\Phi\text{-}1}{=} & \{\!|\, g \mapsto \lambda u_1 u_2.\Phi_u(\{g(x,y)\},\{\epsilon\}) \,|\!\}.u \\ & = & \{\!|\, g \mapsto \lambda u_1 u_2.\Box_{g(x,y)} \,|\!\}.u\end{array}$$

  - $\Delta_\emptyset(\{g(x,y)\}) = \{\epsilon,1,2\}$
  - $\min\limits_{\lhd}(\Delta_\emptyset(\{g(x,y)\})) = \epsilon$

- $N_3 = \lambda g(x,y).(\lambda g(a,z).xz \mid \lambda g(b,z).z)\, y$

$$\begin{array}{rcl}\Psi(N_3) & \overset{\Psi\text{-Abs}}{=} & \lambda u.\Phi_u(\{g(x,y)\},\emptyset)^\sigma \\ & = & \lambda u.(\{\!|\, g \mapsto \lambda u_1 u_2.\Box_{g(x,y)} \,|\!\}.u)^\sigma \\ & = & \lambda u.\{\!|\, g \mapsto \lambda u_1 u_2.(\lambda w.\{\!|\, g \mapsto\ \lambda w_1 w_2.\{\!|\, a \mapsto u_1\, w_2\,;\, b \mapsto w_2 \,|\!\}.w_1 \,|\!\}.w)\, u_2 \,|\!\}.u\end{array}$$

  - $\sigma = \{\Box_{g(x,y)} \mapsto \Psi(((\lambda g(a,z).xz \mid \lambda g(b,z).z)\, y)^{\nu_1})\}$
  - $\Psi((\lambda g(a,z).xz \mid \lambda g(b,z).z)\, y) = (\lambda w.\{\!|\, g \mapsto\ \lambda w_1 w_2.\{\!|\, a \mapsto x\, w_2\,;\, b \mapsto w_2 \,|\!\}.w_1 \,|\!\}.w)\, y$
  - $\nu_1(x) = u_1$, $\nu_1(y) = u_2$ y $\nu_1(z) = z$ para cualquier $z \neq x, y$

- $M_2 = (\lambda g(x,y).(\lambda g(a,z).xz \mid \lambda g(b,z).z)\, y) \quad (g(\lambda x.x, g(a,b)))$

$$\begin{array}{rcl}\Psi(M_2) & \overset{\Psi\text{-App}}{=} & \Psi(\lambda g(x,y).(\lambda g(a,z).xz \mid \lambda g(b,z).z)\, y) \quad \Psi(g(\lambda x.x, g(a,b))) \\ & = & (\lambda u.\{\!|\, g \mapsto \lambda u_1 u_2.(\lambda w.\{\!|\, g \mapsto\ \lambda w_1 w_2.\{\!|\, a \mapsto u_1\, w_2\,;\, b \mapsto w_2 \,|\!\}.w_1 \,|\!\}.w)\, u_2 \,|\!\}.u) \\ & & (g\,(\lambda v.v)\,(g\,a\,b))\end{array}$$

2.4.3. Invariancia sobre términos puros

Antes de pasar a la sección 2.5, en donde probaremos que la traducción dada se comporta de la manera esperada, vamos a mostrar que Ψ se mantiene invariante sobre los términos puros del cálculo lambda. Por supesto, dichos términos satisfacen trivialmente las cláusulas de la Definición 2.4.1. Que la traducción los preserva es razonable, ya que no podría esperarse que éstos se tradujesen a términos más complejos de $\lambda\mathcal{B}_\mathcal{C}$ cuando esto no fuese necesario, teniendo en cuenta que $\lambda\mathcal{B}_c$ es una extensión del λ-cálculo.

Proposición 2.4.1. Sea $M \in \Lambda$. Luego, $\Psi(M) = M$.

Demostración Por inducción en M.

- $M = x \in \mathcal{V}$

Este caso es trivialmente cierto por la ecuación Ψ-VAR.

- $M = M_1\, M_2$

$$\Psi(M) = \Psi(M_1\, M_2) \overset{\Psi\text{-App}}{=} \Psi(M_1)\, \Psi(M_2) \overset{\text{HI}}{=} M_1\, M_2 = M$$

- $M = \lambda x.M_1$, $x \in \mathcal{V}$

$$\begin{array}{rcl}
\Psi(M) & = & \Psi(\lambda x.M_1) \\
& \overset{\Psi\text{-Abs}}{=} & \lambda v.\Phi_v(\{x\},\emptyset)^{\left\{\square_x \mapsto \Psi(M_1^{\{x\mapsto v\}})\right\}} \\
& \overset{\Phi\text{-1}}{=} & \lambda v.\Phi_v(\{x\},\{\epsilon\})^{\left\{\square_x \mapsto \Psi(M_1^{\{x\mapsto v\}})\right\}} \\
& \overset{\Phi\text{-1}}{=} & \lambda v.\square_x^{\left\{\square_x \mapsto \Psi(M_1^{\{x\mapsto v\}})\right\}} \\
& \overset{\text{Sub-Var-Bc}}{=} & \lambda v.\Psi(M_1^{\{x\mapsto v\}}) \\
& \overset{\text{Lema 2.7.9}}{=_\alpha} & \lambda x.\Psi(M_1) \\
& \overset{\text{HI}}{=} & \lambda x.M_1 \\
& = & M
\end{array}$$

□

2.4.4. Otras observaciones sobre Φ

En lo que resta del capítulo, usualmente recurriremos a la prueba de propiedades sobre Φ utilizando el mecanismo de inducción. Para ello, por supuesto, es necesario determinar de antemano sobre qué variable se realizará la inducción, lo cual no está claro a simple vista en este caso particular. No obstante, observemos lo siguiente:

Observación 2.4.1. Asumiendo que $(P, S) \in \mathrm{Dom}(\Phi)$,

(a) En aquellos puntos de la Figura 2.7 en donde $\Phi_v(P, S)$ se define en términos de $\Phi_v(P', S')$,

$$|\bigcup_{p\in P} \mathrm{Pos}(p) \setminus S| > |\bigcup_{p\in P'} \mathrm{Pos}(p) \setminus S'|$$

(b) $|\bigcup_{p\in P} \mathrm{Pos}(p) \setminus S| = 0 \;\Rightarrow\; |P| = 1$

Demostración Para (a), observemos primero que, por definición de Φ, $P' \subseteq P$ y $S' = S \cup \{s\}$, siendo $s = \min_{\lhd}(\Delta_S(P))$. De esta manera, tenemos que $\bigcup_{p\in P'} \mathrm{Pos}(p) \subseteq \bigcup_{p\in P} \mathrm{Pos}(p)$. En caso de igualdad, sabemos que la posición s es tal que $s \in \bigcap_{p\in P} \mathrm{Pos}(p) \subseteq \bigcup_{p\in P} \mathrm{Pos}(p)$, con lo cual $s \in \bigcup_{p\in P} \mathrm{Pos}(p) \setminus S$ pero $s \notin \bigcup_{p\in P'} \mathrm{Pos}(p) \setminus S'$. En otro caso, el resultado es trivial.

Para (b), supongamos que $|P| > 1$. La hipótesis dada implica que $\bigcup_{p\in P} \text{Pos}(p) \subseteq S$, de manera que $\bigcap_{p\in P} \text{Pos}(p) \subseteq S$. Así, $\Delta_S(P) = \emptyset$, de lo cual, junto con que $|P| > 1$, se concluye que $(P, S) \notin \text{Dom}(\Phi)$. No obstante, esto contradice lo que estábamos asumiendo, y la contradicción provino de suponer que $|P| > 1$ [3].

□

Esto nos permite concluir no sólo que cada invocación a Φ desde la función Ψ está bien definida como función total (dado que $>$ es un orden bien fundado sobre $\mathbb{N}$), sino además que es posible hacer inducción sobre el cardinal del conjunto finito $\bigcup_{p\in P} \text{Pos}(p) \setminus S$, tomando como base el caso en el cual dicho valor es cero.

2.5. Simulación

El objetivo de esta sección es demostrar que la traducción propuesta permite simular tanto reducción β-multiple-pattern como η-reducción. Para el primer caso, vamos a mostrar además que un paso de $\to_{\lambda\mathcal{C}}$ se simula en una cantidad linealmente acotada de reducciones en $\lambda\mathcal{B}_c$. Comenzaremos con esta propiedad en la sección 2.5.1 y estudiaremos la η-reducción en la sección 2.5.2.

Los lemas auxiliares que se utilizan aparecen en la sección 2.7.

2.5.1. Reducción β-multiple-pattern

En esta sección, la notación $M \xrightarrow{s} N$ debe entenderse como reducción en un paso a través de la contracción del redex de la posición s en M. Por otra parte, α y α' notarán números naturales representando la cantidad de pasos involucrados en las distintas reducciones.

Proposición 2.5.1. Sean $M, N \in \mathfrak{M}$ y $s \in \mathbb{N}^* \,/\, M \xrightarrow{s}_{\lambda\mathcal{C}} N$. Luego,

$$\Psi(M) \xrightarrow{\alpha\,+}_{\lambda\mathcal{B}_c} \Psi(N)$$

con $\alpha \leq 3\,|p| - 1$ y siendo p el patrón instanciado en el redex de la posición s en M.

Demostración Por inducción en M.

- $M = x \in \mathcal{V}$

  Una variable es $\to_{\lambda\mathcal{C}}$-forma normal, con lo cual la implicación es vacuamente cierta.

- $M = c(M_1, \cdots, M_n)$, $c \in \mathcal{C}$

  Si $M \xrightarrow{s}_{\lambda\mathcal{C}} N$, debe existir $j = 1, \cdots, n$ tal que $M_j \xrightarrow{s'}_{\lambda\mathcal{C}} M'_j$, por lo que $N = c(M_1, \cdots, M'_j, \cdots, M_n)$, con $s = j \cdot s'$. Entonces, a partir de la hipótesis inductiva sobre M_j,

$$\begin{array}{rcl}
\Psi(M) & = & \Psi(c(M_1, \cdots, M_n)) \\
 & \overset{\Psi\text{-PAT}}{=} & c\,\Psi(M_1)\cdots\Psi(M_j)\cdots\Psi(M_n) \\
 & \xrightarrow{\alpha'\,+}_{\lambda\mathcal{B}_c} & c\,\Psi(M_1)\cdots\Psi(M'_j)\cdots\Psi(M_n) \\
 & \overset{\Psi\text{-PAT}}{=} & \Psi(c(M_1, \cdots, M'_j, \cdots, M_n)) \\
 & = & \Psi(N)
\end{array}$$

[3] Notar que P no puede ser vacío pues por hipótesis sabemos que $(P, S) \in \text{Dom}(\Phi)$

Además, $\alpha = \alpha' \leq 3\,|p| - 1$, siendo p el patrón instanciado en el redex de la posición $s = j \cdot s'$.

- $M = \lambda p_1.M_1 \mid \cdots \mid \lambda p_k.M_k$

De manera similar, si $M \xrightarrow{s}_{\lambda\mathcal{C}} N$, debe existir $j = 1, \cdots, k$ tal que $M_j \xrightarrow{s'}_{\lambda\mathcal{C}} M'_j$. Así, $N = \lambda p_1.M_1 \mid \cdots \mid \lambda p_j.M'_j \mid \cdots \mid \lambda p_k.M_k$, con $s = j \cdot 2 \cdot s'$.

Por la ecuación Ψ-Abs, tenemos que $\Psi(M) = \lambda v.\Phi_v\,(\{p_1, \cdots, p_k\}, \emptyset)^\rho$. Ahora bien, por el Lema 2.7.3, la variable $\Box_{p_j}$ aparecerá (una única vez) en $\Phi_v\,(\{p_1, \cdots, p_k\}, \emptyset)$, i.e., en el término $\Phi_v\,(\{p_1, \cdots, p_k\}, \emptyset)^\rho$ aparece (una única vez) el término $\Psi(M_j^{\omega_j})$. Dado que ω_j sólo renombra variables, conserva los tamaños, luego por hipótesis inductiva vale que $\Psi(M_j^{\omega_j}) \xrightarrow{\alpha'+}_{\lambda\mathcal{B}_c} \Psi(M_j'^{\omega_j})$ con $\alpha' \leq 3\,|p| - 1$, siendo p el patrón instanciado en el redex de la posición s'. De esta manera,

$$\begin{array}{rcl}
\Psi(M) & = & \Psi(\lambda p_1.M_1 \mid \cdots \mid \lambda p_k.M_k) \\
& \overset{\Psi\text{-Abs}}{=} & \lambda v.\Phi_v\,(\{p_1, \cdots, p_k\}, \emptyset)^\rho \\
& \xrightarrow{\alpha+}_{\lambda\mathcal{B}_c} & \lambda v.\Phi_v\,(\{p_1, \cdots, p_k\}, \emptyset)^{\rho'} \\
& \overset{\Psi\text{-Abs}}{=} & \Psi(\lambda p_1.M_1 \mid \cdots \mid \lambda p_j.M'_j \mid \cdots \mid \lambda p_k.M_k) \\
& = & \Psi(N)
\end{array}$$

por esta observación, donde $\alpha = \alpha'$, y donde la sustitución ρ' es tal que $\rho'(\Box_{p_j}) = \Psi(M_j'^{\omega_j})$ y $\rho'(x) = \rho(x)$ en cualquier otro caso.

- $M = M_1\,M_2$

En este caso hay que distinguir entre dos subcasos: si la reducción ocurre internamente (i.e., dentro de M_1 o dentro de M_2) o bien si ocurre en la raíz. Si la reducción es interna, y $M_1 \xrightarrow{s'}_{\lambda\mathcal{C}} M'_1 \Rightarrow N = M'_1\,M_2$, con $s = 1 \cdot s'$. Luego,

$$\begin{array}{rcl}
\Psi(M) & = & \Psi(M_1\,M_2) \\
& \overset{\Psi\text{-App}}{=} & \Psi(M_1)\,\Psi(M_2) \\
& \xrightarrow{\alpha'+}_{\lambda\mathcal{B}_c} & \Psi(M'_1)\,\Psi(M_2) \\
& \overset{\Psi\text{-App}}{=} & \Psi(M'_1\,M_2) \\
& = & \Psi(N)
\end{array}$$

donde $\alpha = \alpha' \leq 3\,|p| - 1$ y p es el patrón instanciado en el redex de la posición $s = 1 \cdot s'$.

Si la reducción ocurriese dentro de M_2, el razonamiento es análogo.

Ahora supongamos que la reducción es en la raíz. En este caso, $M_1 = \lambda p_1.M'_1 \mid \cdots \mid \lambda p_k.M'_k$ y, en consecuencia, debe existir $j = 1, \cdots, k$ tal que $M_2 = p_j^\sigma$ para alguna $\lambda\mathcal{C}$-sustitución σ, y $N = M_j'^\sigma$, es decir:

$$M = \left(\lambda p_1.M'_1 \mid \cdots \mid \lambda p_k.M'_k\right)\, p_j^\sigma \xrightarrow{\epsilon}_{\lambda\mathcal{C}} M_j'^\sigma = N$$

Luego,

$$\begin{array}{rcl}
\Psi(M) & = & \Psi(M_1\, M_2) \\
& \overset{\Psi\text{-App}}{=} & \Psi(M_1)\, \Psi(M_2) \\
& \overset{\Psi\text{-Abs}}{=} & (\lambda v.\Phi_v\, (\{p_1, \cdots, p_k\}, \emptyset)^\rho)\ \Psi(p_j^\sigma) \\
& \overset{\text{AppLam}}{\to} & \Phi_v\, (\{p_1, \cdots, p_k\}, \emptyset)^\rho\, [v \,/\, \Psi(p_j^\sigma)]
\end{array}$$

En primer lugar, analicemos qué sucede cuando algún $p_n \in \mathcal{V}$. Por la cláusula 2.4.1*iii* claramente $k = 1 = j$, i.e., $M_1 = \lambda p_1.M_1'$, con $p_1 \in \mathcal{V}$. Luego,

$$\begin{array}{rcl}
\Phi_v\, (\{p_1\}, \emptyset)^\rho\, [v \,/\, \Psi(p_1^\sigma)] & \overset{\Phi\text{-1}}{=} & \Phi_v\, (\{p_1\}, \{\epsilon\})^\rho\, [v \,/\, \Psi(p_1^\sigma)] \\
& \overset{\Phi\text{-1}}{=} & \Box_{p_1}^\rho [v \,/\, \Psi(p_1^\sigma)] \\
& \overset{\text{Sub-Var-Bc}}{=} & \Psi(M_1'^{\omega_1})[v \,/\, \Psi(p_1^\sigma)] \\
& \overset{\text{Lema 2.7.9}}{=} & \Psi\left(M_1'^{\omega_1}[v \,/\, p_1^\sigma]\right) \\
& = & \Psi\left(M_1'[p_1 \,/\, p_1^\sigma]\right) \text{ dado que } \omega_1 = \{p_1 \mapsto v\} \text{ y } v \text{ fresca} \\
& = & \Psi(N)
\end{array}$$

De esta manera, $\alpha = 1 < 2 = 3\,|p_1| - 1$.

Supongamos ahora que ningún patrón es una variable. Por el Lema 2.7.1, sabemos que

$$\begin{array}{rcccll}
 & \Phi_v\, (\{p_1, \cdots, p_k\}, \emptyset)\, [v \,/\, \Psi(p_j)] & \overset{\alpha'}{\twoheadrightarrow}_{\lambda\mathcal{B}_c} & \Box_{p_j} & & \\
\Rightarrow & (\Phi_v\, (\{p_1, \cdots, p_k\}, \emptyset)\, [v \,/\, \Psi(p_j)])^\rho & \overset{\alpha'}{\twoheadrightarrow}_{\lambda\mathcal{B}_c} & \Box_{p_j}^\rho & & \\
\Rightarrow & \Phi_v\, (\{p_1, \cdots, p_k\}, \emptyset)^\rho\, [v \,/\, \Psi(p_j)] & \overset{\alpha'}{\twoheadrightarrow}_{\lambda\mathcal{B}_c} & \Psi(M_j'^{\omega_j}) & & \text{(Lema 2.7.13)} \\
\Rightarrow & \Phi_v\, (\{p_1, \cdots, p_k\}, \emptyset)^\rho\, [v \,/\, \Psi(p_j)^{\tau_j}] & \overset{\alpha'}{\twoheadrightarrow}_{\lambda\mathcal{B}_c} & \Psi(M_j'^{\omega_j})^{\tau_j} & & \text{(Coro. 2.7.15)} \\
\Rightarrow & \Phi_v\, (\{p_1, \cdots, p_k\}, \emptyset)^\rho\, [v \,/\, \Psi(p_j^\sigma)] & \overset{\alpha'}{\twoheadrightarrow}_{\lambda\mathcal{B}_c} & \Psi(M_j'^{\sigma}) & & \text{(Lema 2.7.8)} \\
\Rightarrow & \Psi(M) & \overset{\alpha\ +}{\to}_{\lambda\mathcal{B}_c} & \Psi(N) & &
\end{array}$$

en donde τ_j es la $\lambda\mathcal{B}_c$-sustitución introducida en el Lema 2.7.8:

$$\tau_j(z) = \begin{cases} \Psi(z^{\sigma \circ \omega_j^{-1}}) & \text{si } \exists x \in \mathrm{FV}(p_j) \,/\, \omega_j(x) = z \\ \Psi(z^\sigma) & \text{si } z \in \mathrm{FV}(p_j) \\ z & \text{en otro caso} \end{cases}$$

Se tiene que $\alpha' \leq 3\,|p_j| - 2$ y $\alpha = 1 + \alpha'$, con lo cual $\alpha \leq 3\,|p_j| - 1$.

La utilización de cada Lema queda justificada en cada paso por lo siguiente:

(2.7.13)

- $\mathrm{Dom}(\rho) = \{\Box_{p_1}, \cdots, \Box_{p_k}\} \cap \{v\} = \emptyset$
- $v \notin \mathrm{VRan}(\rho) = \bigcup_{1 \leq i \leq k} \mathrm{FV}(\Psi(M_i'^{\omega_i}))$
- $\mathrm{Dom}(\rho) = \{\Box_{p_1}, \cdots, \Box_{p_k}\} \cap \mathrm{FV}(\Psi(p_j)) = \emptyset$ pues la traducción de un patrón sólo hace uso de las ecuaciones Ψ-Var y Ψ-Pat, que nunca introducen variables $\Box_p$.

(2.7.15) $\mathrm{Dom}(\tau_j) \subseteq \{v_s \in \mathfrak{V} \,/\, \exists\, x \in \mathrm{FV}(p_j) : \omega_j(x) = v_s\} \cup \mathrm{FV}(p_j)$
Las variables $v_s \neq v$, como consecuencia del Corolario 2.7.4, no aparecerán libres en $\Phi_v(\{p_1, \cdots, p_k\}, \emptyset)$, de manera que tampoco estarán en $\Phi_v(\{p_1, \cdots, p_k\}, \emptyset)^\rho$[4]. Además, aquellas variables x presentes en p_j son las que justamente dicho patrón está ligando en M'_j, con lo cual, por convención de Barendregt, se asumen distintas a las restantes variables del término. De esta manera,

$$\mathrm{Dom}(\tau_j) \cap \mathrm{FV}(\Phi_v(\{p_1, \cdots, p_k\}, \emptyset)^\rho) = \emptyset$$

□

2.5.2. η-reducción

En esta subsección nos limitamos a probar que un paso de η-reducción clásica de $\lambda\mathcal{C}$ se traduce en un paso de LamApp como es esperable. De este modo se preserva la posible regla de extensionalidad de ser considerada dentro del cálculo original.

Proposición 2.5.2. Sean $M, N \in \mathfrak{M}$ tales que $M \to_\eta N$. Luego,

$$\Psi(M) \to_\eta \Psi(N)$$

Demostración Por inducción en M.

- $M = x \in \mathcal{V}$

Una variable es $\to_\eta$-forma normal, de manera que la implicación es vacuamente cierta.

- $M = c(M_1, \cdots, M_n)$, $c \in \mathcal{C}$

En este caso, la reducción debe ser interna, i.e., debe existir j tal que $M_j \to_\eta M'_j$, por lo que $N = c(M_1, \cdots, M'_j, \cdots, M_n)$. Luego,

$$\begin{array}{rcl}
\Psi(M) & = & \Psi(c(M_1, \cdots, M_n)) \\
 & \overset{\Psi\text{-PAT}}{=} & c\,\Psi(M_1) \cdots \Psi(M_j) \cdots \Psi(M_n) \\
 & \overset{\mathrm{HI}}{\to_\eta} & c\,\Psi(M_1) \cdots \Psi(M'_j) \cdots \Psi(M_n) \\
 & \overset{\Psi\text{-PAT}}{=} & \Psi(c(M_1, \cdots, M'_j, \cdots, M_n)) \\
 & = & \Psi(N)
\end{array}$$

- $M = M_1\, M_2$

Al igual que en el caso anterior, la reducción debe ser interna. Supongamos que $M_1 \to_\eta M'_1 \Rightarrow N = M'_1\, M_2$. Entonces:

$$\begin{array}{rcl}
\Psi(M) & = & \Psi(M_1\, M_2) \\
 & \overset{\Psi\text{-APP}}{=} & \Psi(M_1)\,\Psi(M_2) \\
 & \overset{\mathrm{HI}}{\to_\eta} & \Psi(M'_1)\,\Psi(M_2) \\
 & \overset{\Psi\text{-APP}}{=} & \Psi(M'_1\, M_2) \\
 & = & \Psi(N)
\end{array}$$

Por otro lado, si la reducción ocurriese dentro de M_2, el razonamiento es análogo.

[4] Esto se debe a que Ψ sólo introduce tales variables en forma ligada a través de Φ, por lo que no aparecerán en el rango de variables de ρ.

- $M = \lambda p_1.M_1 \mid \cdots \mid \lambda p_k.M_k$

En esta oportunidad, debemos considerar por un lado una reducción interna y, por otro, reducción en la raíz. En primera instancia, supongamos que $\exists j \ / \ M_j \to_\eta M'_j \wedge N = \lambda p_1.M_1 \mid \cdots \mid \lambda p_j.M'_j \mid \cdots \mid \lambda p_k.M_k$.

Tenemos que $\Psi(M) \overset{\Psi\text{-Abs}}{=} \Phi_v\left(\{p_1, \cdots, p_k\}, \emptyset\right)^\rho$. El Lema 2.7.3 nos asegura que la variable $\square_{p_j}$ aparecerá (una única vez) en $\Phi_v\left(\{p_1, \cdots, p_k\}, \emptyset\right)$, i.e., en el término $\Phi_v\left(\{p_1, \cdots, p_k\}, \emptyset\right)^\rho$ aparece (una única vez) el término $\Psi(M_j^{\omega_j})$. Dado que ω_j sólo renombra variables, conserva los tamaños, luego por hipótesis inductiva vale que $\Psi(M_j^{\omega_j}) \to_\eta \Psi(M_j'^{\omega_j})$. Así:

$$\begin{array}{rcl}
\Psi(M) & = & \Psi(\lambda p_1.M_1 \mid \cdots \mid \lambda p_k.M_k) \\
& \overset{\Psi\text{-Abs}}{=} & \lambda v.\Phi_v\left(\{p_1, \cdots, p_k\}, \emptyset\right)^\rho \\
& \to_\eta & \lambda v.\Phi_v\left(\{p_1, \cdots, p_k\}, \emptyset\right)^{\rho'} \\
& \overset{\Psi\text{-Abs}}{=} & \Psi(\lambda p_1.M_1 \mid \cdots \mid \lambda p_j.M'_j \mid \cdots \mid \lambda p_k.M_k) \\
& = & \Psi(N)
\end{array}$$

por esta observación. La sustitución ρ' es tal que $\rho'(\square_{p_j}) = \Psi(M_j'^{\omega_j})$ y $\rho'(x) = \rho(x)$ en cualquier otro caso.

Si la reducción se diese en la raíz, debe ocurrir que $k = 1$, $p_1 = x \in \mathcal{V}$ y $M_1 = M'_1\, x$, con $x \notin \mathrm{FV}(M'_1)$. Entonces,

$$M = \lambda x.M'_1\, x \to_\eta M'_1 = N$$

De esta manera,

$$\begin{array}{rcl}
\Psi(M) & = & \Psi(\lambda x.M'_1\, x) \\
& \overset{\Psi\text{-Abs}}{=} & \lambda v.\Phi_v(\{x\}, \emptyset)^{\left\{\square_x \mapsto \Psi((M'_1\, x)^{\{x \mapsto v\}})\right\}} \\
& \overset{\Phi\text{-1}}{=} & \lambda v.\Phi_v(\{x\}, \{\epsilon\})^{\left\{\square_x \mapsto \Psi((M'_1\, x)^{\{x \mapsto v\}})\right\}} \\
& \overset{\Phi\text{-1}}{=} & \lambda v.\square_x^{\left\{\square_x \mapsto \Psi((M'_1\, x)^{\{x \mapsto v\}})\right\}} \\
& \overset{\text{Sub-Var-Bc}}{=} & \lambda v.\Psi\left((M'_1\, x)^{\{x \mapsto v\}}\right) \\
& \overset{\text{Sub-App-C}}{=} & \lambda v.\Psi\left(M_1'^{\{x \mapsto v\}}\, x^{\{x \mapsto v\}}\right) \\
& \overset{x \notin \mathrm{FV}(M'_1)}{=} & \lambda v.\Psi\left(M'_1\, x^{\{x \mapsto v\}}\right) \\
& \overset{\text{Sub-Var-C}}{=} & \lambda v.\Psi(M'_1\, v) \\
& \overset{\Psi\text{-Pat}}{=} & \lambda v.\Psi(M'_1)\, \Psi(v) \\
& \overset{\Psi\text{-Var}}{=} & \lambda v.\Psi(M'_1)\, v \\
& \overset{v \notin \mathrm{FV}(\Psi(M'_1))}{\to_\eta} & \Psi(M'_1) \\
& = & \Psi(N)
\end{array}$$

□

Como se observa en todos los casos, la simulación de η se produce a través de la regla η sin necesidad de apelar a las restantes reglas de $\lambda\mathcal{B}_c$.

2.6. Análisis de tamaños

Toda vez que un lenguaje de programación se propone como alternativa para su uso con respecto a otro, incluso en el caso de prototipos o modelos, son relevantes las cuestiones de complejidad en espacio, esto es, los tamaños de los términos traducidos. En esta sección nos proponemos acotar el tamaño de los términos traducidos en función del tamaño del término original. La experimentación efectuada (ver 2.6.1) sugirió una cota lineal en el peor caso, que es de hecho lo que prueba la Proposición 2.6.1. Primeramente hemos implementado un programa que permite la generación de términos aleatorios, con tamaños y formas variadas. Pruebas empíricas nos han hecho comprobar que la cota que mencionamos en general no suele alcanzarse. La relación entre tamaños, si bien es lineal, determina en la mayoría de los casos prácticos una constante bastante más pequeña.

Otro detalle interesante es que existen casos para los cuales el tamaño del término traducido es estrictamente menor que el tamaño del término original. Éstos son casos donde el término posee abstracciones múltiples donde muchos patrones comparten una estructura similar aunque, desde ya, distinguible. La función de traducción aprovecha esta redundancia estructural generando un case binder sin información repetida, con lo cual se logra una mejora considerable en cuanto al tamaño, sobre todo si el término inicial es razonablemente grande.

Ahora pasaremos a demostrar la proposición principal. Luego mostraremos los resultados obtenidos empíricamente junto con ciertas familias de términos de especial interés debido a sus particulares características.

Proposición 2.6.1. Sea $M \in \mathfrak{M}$. Entonces,

$$|\Psi(M)| \leq 7\,|M| - 1$$

Demostración Por inducción en M.

- $M = x \in \mathcal{V}$

$$|\Psi(M)| = |\Psi(x)| \overset{\Psi\text{-Var}}{=} |x| \overset{\text{S-Var-Bc}}{=} 1 \leq 6 \overset{\text{S-Var-C}}{=} 7\,|x| - 1 = 7\,|M| - 1$$

- $M = M_1\, M_2$

$$\begin{array}{rcl}
|\Psi(M)| & = & |\Psi(M_1\, M_2)| \\
& \overset{\Psi\text{-Pat}}{=} & |\Psi(M_1)\,\Psi(M_2)| \\
& \overset{\text{S-App-Bc}}{=} & 1 + |\Psi(M_1)| + |\Psi(M_2)| \\
& \overset{\text{HI}}{\leq} & 1 + (7\,|M_1| - 1) + (7\,|M_2| - 1) \\
& = & 7\,(|M_1| + |M_2|) - 1 \\
& = & 7\,(1 + |M_1| + |M_2|) - 8 \\
& \overset{\text{S-App-C}}{=} & 7\,|M_1\, M_2| - 8 \\
& < & 7\,|M_1\, M_2| - 1 \\
& = & 7\,|M| - 1
\end{array}$$

- $M = c(M_1, \cdots, M_n)$, $c \in \mathcal{C}$

Este caso lo probaremos por inducción en n. En primer lugar, supongamos que $n = 0$, i.e., c posee aridad nula. De esta manera,

$$|\Psi(M)| = |\Psi(c)| \overset{\Psi\text{-App}}{=} |c| \overset{\text{S-Cons-Bc}}{=} 1 \leq 6 \overset{\text{S-Pat-C}}{=} 7\,|c| - 1 = 7\,|M| - 1$$

Por otro lado, si $n > 0$:

$$\begin{array}{rcl} |\Psi(M)| & = & |\Psi\,(c(M_1, \cdots, M_n))\,| \\ & \overset{\Psi\text{-App}}{=} & |c\,\Psi(M_1) \cdots \Psi(M_n)| \\ & = & |(c\,\Psi(M_1) \cdots \Psi(M_{n-1}))\,\Psi(M_n)| \\ & \overset{\text{S-App-Bc}}{=} & 1 + |c\,\Psi(M_1) \cdots \Psi(M_{n-1})| + |\Psi(M_n)| \end{array}$$

Sea $d \in \mathcal{C}$ tal que $\delta(d) = n - 1$ [5], y sea $N = d(M_1, \cdots, M_{n-1})$. Entonces, por hipótesis inductiva, $|\Psi(N)| \leq 7\,|N| - 1$ y, además, $|\Psi(N)| = |c\,\Psi(M_1) \cdots \Psi(M_{n-1})|$. Por ende,

$$\begin{array}{rcl} 1 + |c\,\Psi(M_1) \cdots \Psi(M_{n-1})| + |\Psi(M_n)| & \overset{\text{HI}}{\leq} & 1 + (7\,|d(M_1, \cdots, M_{n-1})| - 1) + (7\,|M_n| - 1) \\ & = & 7\,(|d(M_1, \cdots, M_{n-1})| + |M_n|) - 1 \\ & \overset{\text{S-Pat-C}}{=} & 7\left(1 + \displaystyle\sum_{i=1}^{n-1} |M_i| + |M_n|\right) - 1 \\ & = & 7\left(1 + \displaystyle\sum_{i=1}^{n} |M_i|\right) - 1 \\ & \overset{\text{S-Pat-C}}{=} & 7\,|c(M_1, \cdots, M_n)| - 1 \\ & = & 7\,|M| - 1 \end{array}$$

- $M = \lambda p_1.M_1 \mid \cdots \mid \lambda p_k.M_k$

En este caso, tenemos que $\Psi(M) = \lambda v.\Phi_v(\{p_1, \cdots, p_k\}, \emptyset)^\rho$, en donde $\rho = \{\Box_{p_1} \mapsto \Psi(M_1^{\omega_1}), \cdots, \Box_{p_k} \mapsto \Psi(M_k^{\omega_k})\}$ y cada ω_i es un renombramiento de variables como se explicó en la sección 2.4. Entonces:

[5] De no existir un tal d en $\mathcal{C}$, podemos asumir que sí existe en forma temporal para los fines de la prueba, y luego descartarlo sin dejar rastro.

$$
\begin{array}{lcl}
|\Psi(M)| & = & |\Psi(\lambda p_1.M_1 \mid \cdots \mid \lambda p_k.M_k)| \\
& \overset{\Psi\text{-Abs}}{=} & |\lambda v.\Phi_v(\{p_1, \cdots, p_k\}, \emptyset)^\rho| \\
& \overset{\text{S-Abs-Bc}}{=} & 2 + |\Phi_v(\{p_1, \cdots, p_k\}, \emptyset)^\rho| \\
& \overset{\text{Lemas 2.7.12 y 2.7.4}}{=} & 2 + |\Phi_v(\{p_1, \cdots, p_k\}, \emptyset)| + \sum_{i=1}^{k} (|\Psi(M_i^{\omega_i})| - 1) \\
& = & 2 - k + |\Phi_v(\{p_1, \cdots, p_k\}, \emptyset)| + \sum_{i=1}^{k} |\Psi(M_i)| \\
& \overset{\text{HI}}{\leq} & 2 - k + |\Phi_v(\{p_1, \cdots, p_k\}, \emptyset)| + \sum_{i=1}^{k} (7\,|M_i| - 1) \\
& = & 2 - 2k + |\Phi_v(\{p_1, \cdots, p_k\}, \emptyset)| + 7 \sum_{i=1}^{k} |M_i| \\
& \overset{\text{Lema 2.7.10}}{\leq} & 2 - 2k + \sum_{i=1}^{k} |\Phi_v(\{p_i\}, \emptyset)| + 7 \sum_{i=1}^{k} |M_i| \\
& \overset{\text{Lema 2.7.11}}{\leq} & 2 - 2k + \sum_{i=1}^{k} (7\,|p_i| - 1) + 7 \sum_{i=1}^{k} |M_i| \\
& = & 2 - 3k + 7 \sum_{i=1}^{k} (|p_i| + |M_i|) \\
& \leq & 6 + 7k + 7 \sum_{i=1}^{k} (|p_i| + |M_i|) \\
& = & 7 \left(1 + k + \sum_{i=1}^{k} (|p_i| + |M_i|)\right) - 1 \\
& \overset{\text{S-Abs-C}}{=} & 7\,|(\lambda p_1.M_1 \mid \cdots \mid \lambda p_k.M_k)| - 1 \\
& = & 7\,|M| - 1
\end{array}
$$

Notar que la quinta igualdad se sostiene por el hecho de que cada ω_i sólo renombra variables, de manera que no se alterará el tamaño de los términos involucrados.

□

2.6.1. Experimentación

Los resultados experimentales consisten, por un lado, en pruebas realizadas con grupos de términos generados aleatoriamente y, además, en el estudio específico de ciertas familias de términos. La experimentación aleatoria se llevó a cabo construyendo una cantidad considerable de términos $\lambda\mathcal{C}$ con tamaños entre 50 y 2000, con distribución aproximadamente uniforme, y luego estudiando la relación con los respectivos tamaños de sus traducciones. La Figura 2.8 muestra el cociente entre el tamaño de la traducción y el término original, para un conjunto aleatorio de 500 términos.

Se observa que la relación entre cada par de términos tiene asociada una constante numérica que no supera a 4,5 en ningún caso, lo cual sugeriría que, en promedio, la constante lineal es aún menor que la estudiada en el desarrollo previo. No obstante, existen casos

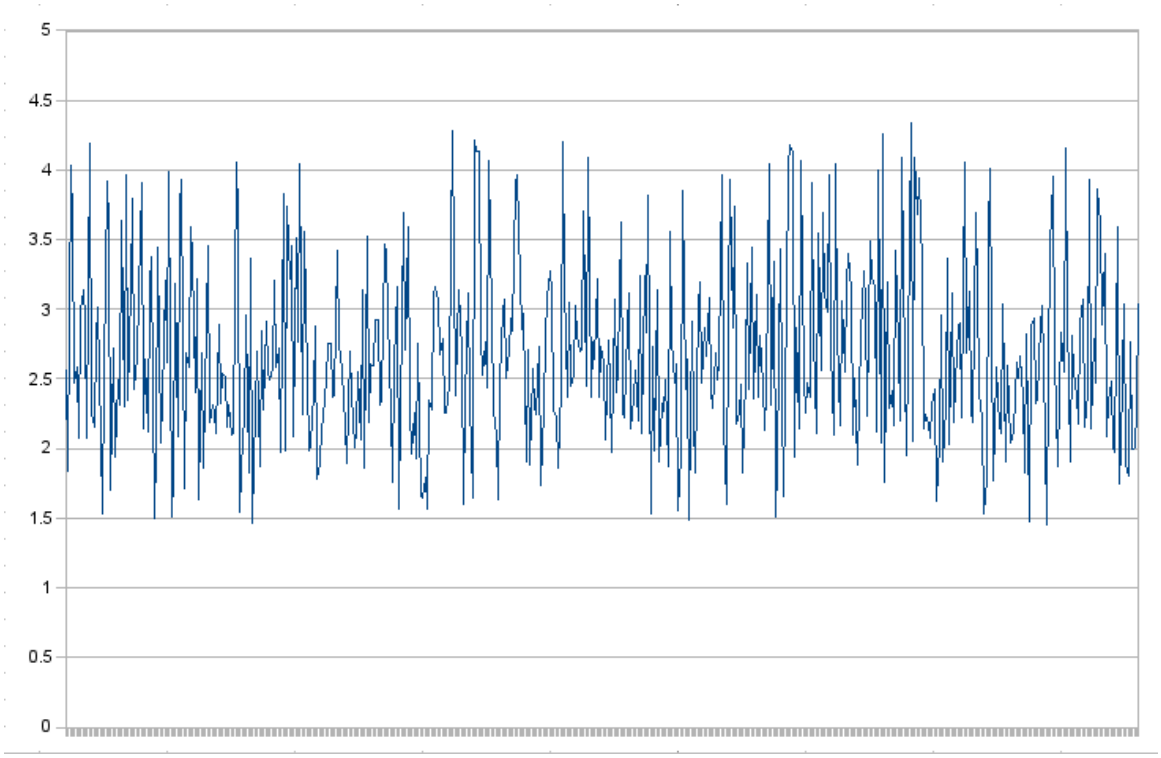

Figura 2.8: Cociente entre tamaños para términos aleatorios y sus traducciones

particulares donde la constante involucrada es exactamente 7. Consideremos, por ejemplo, la familia de términos $\{T_n\}_{n\in\mathbb{N}}$, en donde

$$T_n = \lambda f^n(a).a$$

siendo $f \in \mathcal{C}$ un constructor unario y $a \in \mathcal{C}$ un constructor de aridad nula [6]. Notando que $|f^n(a)| = n + 1$, tenemos que

$$\begin{array}{rcl} |T_n| & = & |\lambda f^n(a).a| \\ & \overset{\text{S-Abs-C}}{=} & 3 + |f^n(a)| \\ & = & n + 4 \end{array}$$

En cuanto a la traducción de T_n,

$$\Psi(T_n) = \Psi(\lambda f^n(a).a) \overset{\Psi\text{-Abs}}{=} \lambda v.\Phi_v(\{f^n(a)\}, \emptyset)^{\{\Box_{f^n(a)} \mapsto a\}}$$

Debido a la estructura del patrón involucrado en la abstracción, los términos que la función Φ generará en cada invocación recursiva no podrán ser otros que los que se muestran debajo, siendo $s = \underset{\lhd}{\text{mín}}(\Delta_S(\{f^n(a)\}))$:

$$\Phi_v(\{f^n(a)\}, S) = \begin{cases} \{\!|\, a \mapsto \Box_{f^n(a)} \,|\!\}.v_s & \text{si } s = 1 \cdots 1 \ (n \text{ veces}) \\ \{\!|\, f \mapsto \lambda v_{s\cdot 1}.\Phi_v(\{f^n(a)\}, S \cup \{s\}) \,|\!\}.v_s & \text{en otro caso} \end{cases}$$

[6] La operación $f^n(M)$ indica la composición de f consigo misma n veces, tomando como término inicial a M.

De esta manera,

$$\begin{array}{rcl} |\Phi_v(\{f^n(a)\},\emptyset)| & = & |\{\!|\, f \mapsto \lambda v_1.\Phi_v(\{f^n(a)\},\{\epsilon\})\,|\!\}.v| \\ & \overset{\text{S-Case-Bc}}{=} & 5 + |\lambda v_1.\Phi_v(\{f^n(a)\},\{\epsilon\})| \\ & \overset{\text{S-Abs-Bc}}{=} & 7 + |\Phi_v(\{f^n(a)\},\{\epsilon\})| \\ & \vdots & \\ & = & 7n + |\{\!|\, a \mapsto \Box_{f^n(a)}\,|\!\}.v_{1\cdots 1}| \\ & \overset{\text{S-Case-Bc}}{=} & 7n + 6 \end{array}$$

Por ende, el tamaño de $\Psi(T_n)$ será

$$\begin{array}{rcl} |\Psi(T_n)| & = & |\lambda v.\Phi_v(\{f^n(a)\},\emptyset)^{\{\Box_{f^n(a)}\mapsto a\}}| \\ & \overset{\text{S-Abs-Bc}}{=} & 2 + |\Phi_v(\{f^n(a)\},\emptyset)^{\{\Box_{f^n(a)}\mapsto a\}}| \\ & \overset{\text{Lemas 2.7.12 y 2.7.3}}{=} & 2 + |\Phi_v(\{f^n(a)\},\emptyset)| \\ & = & 7n + 8 \end{array}$$

En consecuencia, tomando n en función de $|T_n|$ y reemplazando en esta última expresión,

$$|\Psi(T_n)| = 7\,|T_n| - 20$$

Se observa, entonces, que esta familia de términos tiene tamaños que están apenas por debajo de la cota encontrada en la sección anterior.

Una cualidad interesante (y a priori no obvia) de la traducción es que, para ciertos conjuntos de términos $\lambda\mathcal{C}$, su mapeo a $\lambda\mathcal{B}_c$ es tal que el tamaño resultante es estrictamente menor que el original. Esto se da, por ejemplo, en los casos donde el término inicial posee abstracciones múltiples en las que los distintos patrones comparten parte de la estructura. Esta suerte de redundancia estructural es aprovechada naturalmente por $\lambda\mathcal{B}_c$ a través de los case binders, posibilitando de esa manera una reducción en el tamaño.

Definamos, a modo ilustrativo, la familia de términos $\{D_n\}_{n\in\mathbb{N}}$, en donde

$$D_n = \lambda f(x,a_1).a_1 \,|\, \lambda f(x,a_2).a_2 \,|\, \cdots \,|\, \lambda f(x,a_n).a_n$$

siendo $a_1,\cdots,a_n \in \mathcal{C}$ constructores de aridad nula tales que $a_i \neq a_j$ si $1 \leq i < j \leq n$, y siendo $f \in \mathcal{C}$ un constructor binario. Desde ya, D_n satisface la cláusula 2.4.1.

Al tener cada patrón tamaño 3 y cada cuerpo de cada abstracción tamaño 1, la ecuación S-Abs-C nos permite concluir que

$$|D_n| = 5n + 1$$

Por otra parte, para el caso de $\Psi(D_n)$, notemos primero que, por definición de Φ, el símbolo f aparecerá una única vez en el case binder resultante. Luego,

$$
\begin{array}{rcl}
|\Psi(D_n)| & = & |\Psi(\lambda f(x,a_1).a_1 \mid \lambda f(x,a_2).a_2 \mid \cdots \mid \lambda f(x,a_n).a_n)| \\
 & \overset{\Psi\text{-Abs}}{=} & |\lambda v.\Phi_v(\{f(x,a_1),\cdots,f(x,a_n)\},\emptyset)^{\{\Box_{f(x,a_i)} \mapsto a_i\}_{1\leq i\leq n}}| \\
 & = & |\lambda v.\{\!|\, f \mapsto \lambda v_1 v_2.\{\!|\, a_1 \mapsto a_1; \cdots; a_n \mapsto a_n \,|\}.v_2 \,|\}.v| \\
 & \overset{\text{S-Abs-Bc}}{=} & 2 + |\{\!|\, f \mapsto \lambda v_1 v_2.\{\!|\, a_1 \mapsto a_1; \cdots; a_n \mapsto a_n \,|\}.v_2 \,|\}.v| \\
 & \overset{\text{S-Case-Bc}}{=} & 7 + |\lambda v_1 v_2.\{\!|\, a_1 \mapsto a_1; \cdots; a_n \mapsto a_n \,|\}.v_2| \\
 & \overset{\text{S-Abs-Bc}}{=} & 11 + |\{\!|\, a_1 \mapsto a_1; \cdots; a_n \mapsto a_n \,|\}.v_2| \\
 & \overset{\text{S-Case-Bc}}{=} & 11 + (3n+3) \\
 & = & 3n + 14
\end{array}
$$

Combinando ambos resultados, se obtiene que $3n + 14 < 5n + 1$ si y sólo si $n \geq 7$, i.e.,

$$|D_n| > |\Psi(D_n)| \qquad \forall n \in \mathbb{N} \,/\, n \geq 7$$

Este ejemplo permite dar cuenta de manera tangible de la distinta naturaleza de ambos formalismos: los case binders de $\lambda\mathcal{B}_c$, en algunas ocasiones, posibilitan disponer la información más eficientemente que las construcciones de abstracciones múltiples de $\lambda\mathcal{C}$.

2.7. Lemas auxiliares

Pasamos ahora a demostrar los distintos lemas utilizados en las secciones previas así como también otros lemas que éstos requieren.

El primero de ellos es clave en la simulación para el caso de reducción en la raíz, y permite obtener a partir de la expresión de ésta la variable especial que será reemplazada por la traducción del cuerpo de la correspondiente abstracción. A la vez nos da una cota en el número de reducciones.

Lema 2.7.1. Sea $(\{p_1,\cdots,p_k\},\emptyset) \in \mathrm{Dom}(\Phi)$ y $p \in \{p_1,\cdots,p_k\}$. Entonces,

$$\Phi_v(\{p_1,\cdots,p_k\},\emptyset)[v \,/\, \Psi(p)] \overset{r}{\twoheadrightarrow}_{\lambda\mathcal{B}_c} \Box_p$$

con $r \leq 3\,|p| - 2$

Demostración Como consecuencia de la Observación 2.4.1, $\Phi_v(\{p_1,\cdots,p_k\},\emptyset)$ finalizará en una cantidad finita de pasos (donde cada paso representa una invocación recursiva). Por ende, sea $l \in \mathbb{N}$ la cantidad de invocaciones recursivas realizadas por $\Phi_v(\{p_1,\cdots,p_k\},\emptyset)$ tal que, para cada $j = 0,\cdots,l$, P_j y S_j representan los argumentos de Φ_v y además $p \in P_j$. Tomando $P_0 = \{p_1,\cdots,p_k\}$, dentro de la secuencia de conjuntos de patrones $P_0,\cdots,P_l$ se distinguen dos subsecuencias $P_0,\cdots,P_{n-1}$ y $P_n,\cdots,P_l$ $(0 \leq n \leq l)$ en donde $|P_j| > 1$ si $0 \leq j < n$ y $|P_j| = 1$ en otro caso. De esta manera,

- $\begin{cases} P_0 = \{p_1,\cdots,p_k\} & \\ P_j \in P_{j-1}/\mathcal{R}_{s_{j-1}} \wedge p \in P_j & \text{si } 1 \leq j < n \\ P_j = P_{j-1} = \{p\} & \text{si } n \leq j \leq l \end{cases}$
- $\begin{cases} S_0 = \emptyset & \\ S_j = S_{j-1} \cup \{s_{j-1}\} & \text{para cada } j = 1,\cdots,l \end{cases}$

- $s_j = \min_{\lhd}(\Delta_{S_j}(P_j))$ para cada $j = 0, \cdots, l-1$

Definamos además lo siguiente:

- $c^j_{i_j} = \boldsymbol{c}(p|_{s_j})$ para cada $j = 0, \cdots, n-1$
  i_j es tal que $1 \leq i_j \leq m_j$, siendo m_j la cantidad de clases de equivalencia de P_j módulo $\mathcal{R}_{s_j}$
- Para cada $j = 1, \cdots, l$
  $$\sigma_j(x) = \begin{cases} \Psi(p|_t) & \text{si } x = v_t, \text{ con } t \in \bigcup_{w \in S_j \,/\, p|_w \notin \mathcal{V}} \{w \cdot 1, \cdots, w \cdot \delta(\boldsymbol{c}(p|_w))\} \\ x & \text{en otro caso} \end{cases}$$
- $\sigma_0(x) = \begin{cases} \Psi(p) & \text{si } x = v \\ x & \text{en otro caso} \end{cases}$

En primera instancia, mostraremos que

$$\Phi_v(P_j, S_j)^{\sigma_j} \twoheadrightarrow \Phi_v(P_{j+1}, S_{j+1})^{\sigma_{j+1}} \qquad (0 \leq j < n)$$

$$\begin{array}{lll}
\Phi_v(P_j, S_j)^{\sigma_j} & \overset{\Phi\text{-}2}{=} & \left(\{\!|c^j_1 \mapsto T^j_1; \cdots; c^j_{m_j} \mapsto T^j_{m_j}|\!\}.v_{s_j}\right)^{\sigma_j} \\
& \overset{\text{Sub-Case-Bc}}{=} & \{\!|c^j_1 \mapsto T^{j^{\sigma_j}}_1; \cdots; c^j_{m_j} \mapsto T^{j^{\sigma_j}}_{m_j}|\!\}.v^{\sigma_j}_{s_j} \\
& \overset{\text{Lema 2.7.2}}{=} & \{\!|c^j_1 \mapsto T^{j^{\sigma_j}}_1; \cdots; c^j_{m_j} \mapsto T^{j^{\sigma_j}}_{m_j}|\!\}.\Psi(p|_{s_j}) \\
& \overset{\Psi\text{-Pat}}{=} & \{\!|c^j_1 \mapsto T^{j^{\sigma_j}}_1; \cdots; c^j_{m_j} \mapsto T^{j^{\sigma_j}}_{m_j}|\!\}.\left(c^j_{i_j}\, \Psi(p|_{s_j \cdot 1}) \cdots \Psi(p|_{s_j \cdot \delta(c^j_{i_j})})\right) \\
& \overset{\delta(c^j_{i_j})}{\twoheadrightarrow} & \left(\{\!|c^j_1 \mapsto T^{j^{\sigma_j}}_1; \cdots; c^j_{m_j} \mapsto T^{j^{\sigma_j}}_{m_j}|\!\}.c^j_{i_j}\right)\, \Psi(p|_{s_j \cdot 1}) \cdots \Psi(p|_{s_j \cdot \delta(c^j_{i_j})}) \\
& \to & T^{j^{\sigma_j}}_{i_j}\, \Psi(p|_{s_j \cdot 1}) \cdots \Psi(p|_{s_j \cdot \delta(c^j_{i_j})}) \\
& = & \left(\lambda v_{s_j \cdot 1} \cdots v_{s_j \cdot \delta(c^j_{i_j})}.\Phi_v(P_{j+1}, S_j \cup \{s_j\})\right)^{\sigma_j}\, \Psi(p|_{s_j \cdot 1}) \cdots \Psi(p|_{s_j \cdot \delta(c^j_{i_j})}) \\
& \overset{\text{Sub-Abs-Bc}}{=} & \left(\lambda v_{s_j \cdot 1} \cdots v_{s_j \cdot \delta(c^j_{i_j})}.\Phi_v(P_{j+1}, S_{j+1})^{\sigma_j}\right)\, \Psi(p|_{s_j \cdot 1}) \cdots \Psi(p|_{s_j \cdot \delta(c^j_{i_j})}) \\
& \overset{\delta(c^j_{i_j})}{\twoheadrightarrow} & \Phi_v(P_{j+1}, S_{j+1})^{\sigma_j}[v_{s_j \cdot 1} \,/\, \Psi(p|_{s_j \cdot 1})] \cdots [v_{s_j \cdot \delta(c^j_{i_j})} \,/\, \Psi(p|_{s_j \cdot \delta(c^j_{i_j})})] \\
& \overset{\text{Lema 2.7.2}}{=} & \Phi_v(P_{j+1}, S_{j+1})^{\sigma_{j+1}}
\end{array}$$

Es preciso remarcar que la última igualdad no será justificada por el Lema 2.7.2 si $j = 0$. Sin embargo, el Lema 2.7.3 nos asegura que, en tal caso, σ_0 no tendrá efecto sobre el término $\Phi_v(P_1, S_1)$: $v_{s_0} = v_\epsilon = v$ no aparecerá entre sus variables libres. Por lo tanto,

$$\begin{array}{rcl}
\Phi_v(P_1, S_1)^{\sigma_0}[v_1 \,/\, \Psi(p|_1)] \cdots [v_{\delta(c^0_{i_0})} \,/\, \Psi(p|_{\delta(c^0_{i_0})})] & = & \Phi_v(P_1, S_1) \\
& & [v_1 \,/\, \Psi(p|_1)] \cdots [v_{\delta(c^0_{i_0})} \,/\, \Psi(p|_{\delta(c^0_{i_0})})] \\
& = & \Phi_v(P_1, S_1)^{\sigma_1}
\end{array}$$

Ahora probaremos lo mismo para cuando $n \leq j < l$. En principio observemos que, siempre que s_j sea tal que $p|_{s_j} \in \mathcal{V}$, por la ecuación Φ-1 y por definición de σ_j, tendremos que

$$\Phi_v(P_j, S_j)^{\sigma_j} = \Phi_v(P_j, S_j \cup \{s_j\})^{\sigma_j} = \Phi_v(P_{j+1}, S_{j+1})^{\sigma_j} = \Phi_v(P_{j+1}, S_{j+1})^{\sigma_{j+1}}$$

En consecuencia, nos concentraremos en el caso donde $p|_{s_j} \notin \mathcal{V}$:

$$\begin{array}{lll}
\Phi_v(P_j, S_j)^{\sigma_j} & \overset{\Phi\text{-}1}{=} & \Big(\{\!|\boldsymbol{c}(p|_{s_j}) \mapsto \lambda v_{s_j\cdot 1} \cdots v_{s_j\cdot\delta(\boldsymbol{c}(p|_{s_j}))}.\Phi_v(P_j, S_j \cup \{s_j\})|\!\}.v_{s_j}\Big)^{\sigma_j} \\
& \overset{\text{Sub-Case-Bc}}{=} & \{\!|\boldsymbol{c}(p|_{s_j}) \mapsto \Big(\lambda v_{s_j\cdot 1} \cdots v_{s_j\cdot\delta(\boldsymbol{c}(p|_{s_j}))}.\Phi_v(P_j, S_j \cup \{s_j\})\Big)^{\sigma_j}|\!\}.v_{s_j}^{\sigma_j} \\
& \overset{\text{Sub-Abs-Bc}}{=} & \{\!|\boldsymbol{c}(p|_{s_j}) \mapsto \lambda v_{s_j\cdot 1} \cdots v_{s_j\cdot\delta(\boldsymbol{c}(p|_{s_j}))}.\Phi_v(P_j, S_j \cup \{s_j\})^{\sigma_j}|\!\}.v_{s_j}^{\sigma_j} \\
& \overset{\text{Lema 2.7.2}}{=} & \{\!|\boldsymbol{c}(p|_{s_j}) \mapsto \lambda v_{s_j\cdot 1} \cdots v_{s_j\cdot\delta(\boldsymbol{c}(p|_{s_j}))}.\Phi_v(P_j, S_j \cup \{s_j\})^{\sigma_j}|\!\}.\Psi(p|_{s_j}) \\
& \overset{\Psi\text{-Pat}}{=} & \{\!|\boldsymbol{c}(p|_{s_j}) \mapsto \lambda v_{s_j\cdot 1} \cdots v_{s_j\cdot\delta(\boldsymbol{c}(p|_{s_j}))}.\Phi_v(P_j, S_j \cup \{s_j\})^{\sigma_j}|\!\}. \\
& & \Big(\boldsymbol{c}(p|_{s_j})\Psi(p|_{s_j\cdot 1}) \cdots \Psi(p|_{s_j\cdot\delta(\boldsymbol{c}(p|_{s_j}))})\Big) \\
& \overset{\delta(\boldsymbol{c}(p|_{s_j}))}{\twoheadrightarrow} & \Big(\{\!|\boldsymbol{c}(p|_{s_j}) \mapsto \lambda v_{s_j\cdot 1} \cdots v_{s_j\cdot\delta(\boldsymbol{c}(p|_{s_j}))}.\Phi_v(P_j, S_j \cup \{s_j\})^{\sigma_j}|\!\}.\boldsymbol{c}(p|_{s_j})\Big) \\
& & \Psi(p|_{s_j\cdot 1}) \cdots \Psi(p|_{s_j\cdot\delta(\boldsymbol{c}(p|_{s_j}))}) \\
& \to & \Big(\lambda v_{s_j\cdot 1} \cdots v_{s_j\cdot\delta(\boldsymbol{c}(p|_{s_j}))}.\Phi_v(P_j, S_j \cup \{s_j\})^{\sigma_j}\Big)\ \Psi(p|_{s_j\cdot 1}) \cdots \Psi(p|_{s_j\cdot\delta(\boldsymbol{c}(p|_{s_j}))}) \\
& \overset{\delta(\boldsymbol{c}(p|_{s_j}))}{\twoheadrightarrow} & \Phi_v(P_j, S_j \cup \{s_j\})^{\sigma_j}[v_{s_j\cdot 1} \,/\, \Psi(p|_{s_j\cdot 1})] \cdots [v_{s_j\cdot\delta(\boldsymbol{c}(p|_{s_j}))} \,/\, \Psi(p|_{s_j\cdot\delta(\boldsymbol{c}(p|_{s_j}))})] \\
& \overset{\text{Lema 2.7.2}}{=} & \Phi_v(P_j, S_j \cup \{s_j\})^{\sigma_{j+1}} \\
& = & \Phi_v(P_{j+1}, S_{j+1})^{\sigma_{j+1}}
\end{array}$$

La observación sobre la igualdad justificada mediante el Lema 2.7.2 realizada anteriormente también se aplica en esta oportunidad.

Combinando estos resultados, obtenemos lo siguiente:

$$\begin{array}{lll}
\Phi_v(\{p_1, \cdots, p_k\}, \emptyset)[v \,/\, \Psi(p)] & = & \Phi_v(P_0, S_0)^{\sigma_0} \\
& \overset{r_0}{\twoheadrightarrow} & \Phi_v(P_1, S_1)^{\sigma_1} \\
& \vdots & \\
& \overset{r_{n-1}}{\twoheadrightarrow} & \Phi_v(P_n, S_n)^{\sigma_n} \\
& \overset{r_n}{\twoheadrightarrow} & \Phi_v(P_{n+1}, S_{n+1})^{\sigma_{n+1}} \\
& \vdots & \\
& \overset{r_{l-1}}{\twoheadrightarrow} & \Phi_v(P_l, S_l)^{\sigma_l} \\
& = & \square_p^{\sigma_l} \\
& = & \square_p
\end{array}$$

Sea $r = \sum_{j=0}^{l-1} r_j$, i.e., la cantidad total de reducciones. Resta probar que $r \leq 3|p| - 2$. En

base al desarrollo previo, notemos que, para cada $j = 0, \cdots, l-1$,

$$r_j = \begin{cases} 2\delta(\boldsymbol{c}(p|_{s_j})) + 1 & \text{si } p|_{s_j} \notin \mathcal{V} \\ 0 & \text{en caso contrario} \end{cases}$$

De esta manera, el valor máximo que r puede tomar ocurrirá cuando p no posea variables libres (es decir, cuando se trate de un patrón ground). Luego,

$$\begin{aligned} r &\leq \sum_{w \in \text{Pos}(p)} 2\delta(\boldsymbol{c}(p|_{s_j})) + 1 \\ &= 2 \sum_{w \in \text{Pos}(p)} \delta(\boldsymbol{c}(p|_{s_j})) + \sum_{w \in \text{Pos}(p)} 1 \\ &\overset{\text{Lema 2.7.7}}{=} 2(|p| - 1) + |\text{Pos}(p)| \\ &= 2(|p| - 1) + |p| \\ &= 3|p| - 2 \end{aligned}$$

□

Este resultado se utiliza en forma auxiliar al Lema previo.

Lema 2.7.2. En el contexto del Lema 2.7.1, y siendo p no variable,

- $v_{s_j} \in \text{Dom}(\sigma_j) \;\; \forall j = 0, \cdots, l-1$
- $\sigma_{j+1} = \tau \circ \sigma_j \;\; \forall j = 1, \cdots, l-1$
  siendo $\tau = \{v_{s_j \cdot \delta(\boldsymbol{c}(p|_{s_j}))} \mapsto \Psi(p|_{s_j \cdot \delta(\boldsymbol{c}(p|_{s_j}))})\} \circ \cdots \circ \{v_{s_j \cdot 1} \mapsto \Psi(p|_{s_j \cdot 1})\}$

Demostración Para la primera parte, primero notemos que, si $j = 0$, el resultado es trivial: $v_{s_0} = v_\epsilon = v$, siendo v la única variable en $\text{Dom}(\sigma_0)$ [7].

Si $j > 0$, debemos probar que $s_j \in \bigcup_{w \in S_j \,/\, p|_w \notin \mathcal{V}} \{w \cdot 1, \cdots, w \cdot \delta(\boldsymbol{c}(p|_w))\}$. Por definición, $s_j = \min_{\lhd} (\Delta_{S_j}(P_j)) \neq \epsilon$. Por Lema 2.7.5, cada prefijo propio de s_j estará presente en S_j. En particular, dado que $s_j = t \cdot i$, $t \in S_j$ y, además, $i \in \{1, \cdots, \delta(\boldsymbol{c}(p|_t))\}$. Se concluye que $s_j \in \bigcup_{w \in S_j \,/\, p|_w \notin \mathcal{V}} \{w \cdot 1, \cdots, w \cdot \delta(\boldsymbol{c}(p|_w))\}$.

Para la segunda parte, probaremos la doble inclusión de los dominios de las respectivas sustituciones. No obstante, observemos antes que, en el caso de la sustitución $\tau \circ \sigma_j$, ocurre que $\text{Dom}(\tau \circ \sigma_j) = \text{Dom}(\tau) \cup \text{Dom}(\sigma_j)$.

- $\text{Dom}(\sigma_{j+1}) \subseteq \text{Dom}(\tau \circ \sigma_j)$

Sea $v_t \in \text{Dom}(\sigma_{j+1})$. Luego, debe existir $w \in S_{j+1}$ tal que $t = w \cdot i$, con $i \in \{1, \cdots, \delta(\boldsymbol{c}(p|_w))\}$. Dado que $S_{j+1} = S_j \cup \{s_j\}$, puede ocurrir que $w \in S_j$ o bien que $w = s_j$. En el primer caso, trivialmente sigue que $v_t \in \text{Dom}(\sigma_j)$, lo cual implica que $v_t \in \text{Dom}(\tau \circ \sigma_j)$. Si, por otro lado, $w = s_j$, entonces $t = w \cdot i = s_j \cdot i \in \text{Dom}(\tau)$ por lo que, de igual manera, $v_t \in \text{Dom}(\tau \circ \sigma_j)$.

- $\text{Dom}(\tau \circ \sigma_j) \subseteq \text{Dom}(\sigma_{j+1})$

[7] Independiemente del conjunto de patrones involucrado, $s_0 = \epsilon$ en cualquier caso: $\epsilon \in \text{Pos}(p)$ para cualquier $p \in \mathcal{P}$ y, además, $\epsilon \lhd s$ para cualquier $s \in \mathbb{N}^*$

Sea $v_t \in \mathrm{Dom}(\tau \circ \sigma_j)$. Si $v_t \in \mathrm{Dom}(\sigma_j)$, debe existir $w \in S_j$ tal que $t = w \cdot i$, con $i \in \{1, \cdots, \delta(\boldsymbol{c}(p|_w))\}$. Como $S_j \subset S_{j+1}$, tal w también estará en S_{j+1}. En conclusión, $v_t \in \mathrm{Dom}(\sigma_{j+1})$. Si $v_t \in \mathrm{Dom}(\tau)$, entonces $t = s_j \cdot i$, con $i \in \{1, \cdots, \delta(\boldsymbol{c}(p|_{s_j}))\}$. Pero $s_j \in S_{j+1}$ y, luego, sigue que $v_t \in \mathrm{Dom}(\sigma_{j+1})$ por definición.

□

El próximo Lema trata sobre las variables libres de los términos generados por la función Φ.

Lema 2.7.3. Si $(P, S) \in \mathrm{Dom}(\Phi)$ y $S \subseteq \bigcap_{p \in P} \mathrm{Pos}(p)$,

$$\mathrm{FV}(\Phi_v(P,S)) \subseteq \begin{cases} \{\Box_p \,/\, p \in P\} \cup \{v_t \,/\, t = s \vee t \in \mathfrak{H}(P,S)\} & \text{si } \Delta_S(P) \neq \emptyset \\ \{\Box_p\} & \text{en otro caso, siendo } P = \{p\} \end{cases}$$

$$\text{donde } \begin{cases} s = \min_{\lhd}(\Delta_S(P)) \\ \mathfrak{H}(P,S) = \bigcup_{w \in S} \{w \cdot j \,/\, \exists p \in P : p|_w \notin \mathcal{V} \wedge 1 \leq j \leq \delta(\boldsymbol{c}(p|_s))\} \end{cases}$$

Más aún, cada variable $\Box_p$ aparece exactamente una vez en dicho término.

Demostración Por inducción en $|\bigcup_{p \in P} \mathrm{Pos}(p) \setminus S|$.

Si dicho valor es cero, por la Observación 2.4.1, debe ocurrir que $P = \{p\}$ y, además, $\mathrm{Pos}(p) \subseteq S$. Luego, $\Delta_S(P) = \emptyset$, y entonces

$$\mathrm{FV}(\Phi_v(P,S)) \overset{\Phi\text{-}1}{=} \mathrm{FV}(\Box_p) \overset{\text{FV-VAR}}{=} \{\Box_p\}$$

lo cual prueba lo buscábamos.

Si $|\bigcup_{p \in P} \mathrm{Pos}(p) \setminus S| > 0$, necesariamente ocurrirá que $\Delta_S(P) \neq \emptyset$. Debemos analizar los siguientes casos:

- $P = \{p\} \wedge p|_s \in \mathcal{V}$

Sea $s' = \min_{\lhd}(\Delta_{S \cup \{s\}}(P))$.

$$\begin{aligned} \mathrm{FV}(\Phi_v(P,S)) &\overset{\Phi\text{-}1}{=} \mathrm{FV}(\Phi_v(P, S \cup \{s\})) \\ &\overset{\text{HI}}{\subseteq} \begin{cases} \{\Box_p\} \cup \{v_t \,/\, t = s' \vee t \in \mathfrak{H}(P, S \cup \{s\})\} & \text{si } \Delta_{S \cup \{s\}}(P) \neq \emptyset \\ \{\Box_p\} & \text{en otro caso} \end{cases} \end{aligned}$$

Debemos ver que cada variable x perteneciente a estos conjuntos estará presente también en $\{\Box_p\} \cup \{v_t \,/\, t = s \vee t \in \mathfrak{H}(P,S)\}$. Analicemos qué ocurre cuando $x = v_t$, dado que el caso de $x = \Box_p$ es trivial. En primer lugar, supongamos que $t = s' \neq \epsilon$. Por el Lema 2.7.5, cada prefijo propio de t pertenece a $S \cup \{s\}$. En particular, siendo $t = t' \cdot i$, $t' \in S \cup \{s\}$. Pero $t' \neq s$ pues s es posición de variable en p, de manera que $t' \cdot i$ no será una posición válida en dicho patrón. Así, $t' \in S$ y, en consecuencia, $t \in \mathfrak{H}(P,S)$.

Por otro lado, si $t \in \mathfrak{H}(P, S \cup \{s\})$, $\exists w \in S \cup \{s\} \,/\, p|_w \notin \mathcal{V} \wedge t = w \cdot i$, con $1 \leq i \leq \delta(\boldsymbol{c}(p|_w))$. Claramente $w \neq s$ pues $p|_s \in \mathcal{V}$. Luego, $w \in S$, por lo que $t \in \mathfrak{H}(P,S)$.

En cualquiera de estos casos, $t \in \mathfrak{H}(P,S) \Rightarrow x = v_t \in \{v_t \,/\, t = s \vee t \in \mathfrak{H}(P,S)\}$.

- $P = \{p\} \wedge p|_s \notin \mathcal{V}$

Siendo $n = \delta(\boldsymbol{c}(p|_s))$ y $s' = \min_{\lhd}(\Delta_{S\cup\{s\}}(P))$., tenemos que

$$\begin{array}{rcl} \mathrm{FV}(\Phi_v(P,S)) & \overset{\text{Φ-1}}{=} & \mathrm{FV}\left(\{\!|\boldsymbol{c}(p|_s) \mapsto \lambda v_{s\cdot 1}\cdots v_{s\cdot n}.\Phi_v(P, S\cup\{s\})|\!\}.v_s\right) \\ & \overset{\text{FV-Case}}{=} & \mathrm{FV}\left(\lambda v_{s\cdot 1}\cdots v_{s\cdot n}.\Phi_v(P, S\cup\{s\})|\!\}\right) \cup \mathrm{FV}(v_s) \\ & \overset{\text{FV-Abs, FV-Var}}{=} & \mathrm{FV}(\Phi_v(P, S\cup\{s\})) \setminus \{v_{s\cdot 1}, \cdots, v_{s\cdot n}\} \cup \{v_s\} \end{array}$$

Por hipótesis inductiva,

$$\mathrm{FV}(\Phi_v(P, S\cup\{s\})) \subseteq \begin{cases} \{\Box_p\} \cup \{v_t \,/\, t = s' \vee t \in \mathfrak{H}(P, S\cup\{s\})\} & \text{si } \Delta_{S\cup\{s\}}(P) \neq \emptyset \\ \{\Box_p\} & \text{en otro caso} \end{cases}$$

Sea $x \in \mathrm{FV}(\Phi_v(P,S))$. Por el desarrollo previo, debe valer que $x = v_s$ o bien que $x \in \mathrm{FV}(\Phi_v(P, S\cup\{s\}))$ y $x \neq v_{s\cdot i}$, $1 \leq i \leq n$. Veamos que, de cualquier manera, $x \in \{\Box_p\} \cup \{v_t \,/\, t = s \vee t \in \mathfrak{H}(P,S)\}$.

Para $x = v_s$ o $x = \Box_p$, es trivial. Puede también ocurrir que $x = v_t$, con $t = s'$ o $t \in \mathfrak{H}(P, S\cup\{s\})$. En el primer caso, debe suceder que $t \neq s\cdot i$, $1 \leq i \leq n$. Es decir, si $t = t'\cdot j$, entonces $t' \neq s$. Por el Lema 2.7.5 más esta observación, $t' \in S$, lo cual implica que $t \in \mathfrak{H}(P,S)$.

Si $t \in \mathfrak{H}(P, S\cup\{s\})$, debe existir $w \in S\cup\{s\}$ tal que $p|_w \notin \mathcal{V}$ y $t = w\cdot i$, con $1 \leq i \leq \delta(\boldsymbol{c}(p|_w))$. Pero $w \neq s$ porque de no ser así, $t = s\cdot i$, y cada una de las variables $v_{s\cdot i}$ queda excluida del conjunto. Entonces, $w \in S$ y esto implica que $t \in \mathfrak{H}(P,S)$.

De cualquier manera, se obtiene que $x \in \{\Box_p\} \cup \{v_t \,/\, t = s \vee t \in \mathfrak{H}(P,S)\}$, lo cual prueba este caso.

- $P = \{p_1, \cdots, p_k\}$, con $k > 1$

Siendo $n = \delta(\boldsymbol{c}(p|_s))$, tenemos que

$$\begin{array}{rcl} \mathrm{FV}(\Phi_v(P,S)) & \overset{\text{Φ-2}}{=} & \mathrm{FV}\left(\{\!|c_1 \mapsto T_1; \cdots; c_m \mapsto T_m|\!\}.v_s\right) \\ & \overset{\text{FV-Case}}{=} & \displaystyle\bigcup_{i=1}^{m} \mathrm{FV}(T_i) \cup \mathrm{FV}(v_s) \\ & \overset{\text{FV-Var}}{=} & \displaystyle\bigcup_{i=1}^{m} \mathrm{FV}(\lambda v_{s\cdot 1}\cdots v_{s\cdot\delta(c_i)}.\Phi_v(P_i, S\cup\{s\})) \cup \{v_s\} \\ & \overset{\text{FV-Abs}}{=} & \displaystyle\bigcup_{i=1}^{m} \left(\mathrm{FV}(\Phi_v(P_i, S\cup\{s\})) \setminus \{v_{s\cdot 1}, \cdots, v_{s\cdot\delta(c_i)}\}\right) \cup \{v_s\} \end{array}$$

Sea $x \in \mathrm{FV}(\Phi_v(P,S))$. Por lo anterior, debe ocurrir que $x = v_s$ o bien que existe un i_0, $1 \leq i_0 \leq m$, tal que $x \in \mathrm{FV}(\Phi_v(P_{i_0}, S\cup\{s\}))$ y $x \neq v_{s\cdot j}$, $1 \leq j \leq \delta(c_{i_0})$. Probaremos que en cualquier caso, $x \in \{\Box_p \,/\, p \in P\} \cup \{v_t \,/\, t = s \vee t \in \mathfrak{H}(P,S)\}$.

Al igual que antes, el caso de $x = v_s$ es trivial. De la otra manera, por hipótesis inductiva se tiene que

$$x \in \{\Box_p \,/\, p \in P_{i_0}\} \vee x \in \{v_t \,/\, t = s' \vee t \in \mathfrak{H}(P_{i_0}, S\cup\{s\})\}$$

siendo $s' = \min_{\lhd}(\Delta_{S\cup\{s\}}(P_{i_0}))$.

Si $x = \Box_p$ para algún $p \in P_{i_0}$, entonces $x \in \{\Box_p \,/\, p \in P\}$ dado que $P_{i_0} \subseteq P$. Por otra

parte, supongamos que $x = v_t$ y $t = s'$. Por el Lema 2.7.5, cada prefijo propio de t estará en el conjunto $S \cup \{s\}$. Siendo $t = t' \cdot i$, en particular $t' \in S \cup \{s\}$. No puede suceder que $t' = s$ pues, por hipótesis, x no puede ser de la forma $v_{s \cdot j}$, con $1 \leq j \leq \delta(c_{i_0})$. Luego, $t' \in S$, de lo que sigue que $t \in \mathfrak{H}(P, S)$.

Si $x = v_t$ y $t \in \mathfrak{H}(P_{i_0}, S \cup \{s\})$, debe existir una posición w en $S \cup \{s\}$ y un patrón p en P_{i_0} tales que $p|_w \notin \mathcal{V}$ y $t = w \cdot j$, con $1 \leq j \leq \delta(\boldsymbol{c}(p|_w))$. Nuevamente, $w \neq s$ dado que $x = v_t = v_{w \cdot j}$ no puede ser de la forma $v_{s \cdot i}$ por hipótesis. De esta forma, $w \in S$ lo cual implica que $t \in \mathfrak{H}(P, S)$.

Así, se obtiene que $x \in \{\Box_p \,/\, p \in P\} \cup \{v_t \,/\, t = s \vee t \in \mathfrak{H}(P, S)\}$, lo cual prueba este caso. Notar además que cada variable $\Box_p$, $p \in P$, aparecerá exactamente una vez en $\Phi_v(P, S)$: esto es consecuencia de la hipótesis inductiva y del hecho que $\{P_1, \cdots, P_m\}$ es una partición de P.

□

El siguiente es un corolario inmediato del Lema anterior, de interés en el marco de la traducción, donde las invocaciones a Φ se dan con $S = \emptyset$.

Corolario 2.7.4. Sea $(P, \emptyset) \in \mathrm{Dom}(\Phi)$. Luego,

$$\mathrm{FV}(\Phi_v(P, \emptyset)) \subseteq \{\Box_p \,/\, p \in P\} \cup \{v_s\}$$

siendo $s = \min_{\lhd}(\Delta_\emptyset(P))$. Más aún, cada variable $\Box_p$ aparece una única vez en $\Phi_v(P, \emptyset)$.

Demostración Basta instanciar el Lema 2.7.3 con $S = \emptyset$ y notar que, en tal caso, $\mathfrak{H}(P, S) = \emptyset$. □

El próximo resultado se utiliza fuertemente en los lemas previos. En esencia, dice que, dados P y S, cada prefijo propio del mínimo elemento de $\Delta_S(P)$ está contenido en S. Intuitivamente, este resultado es razonable pues dichos prefijos son menores según $\lhd$, de manera que, de no estar en S alguno de ellos, sería el mínimo del conjunto según $\lhd$.

Lema 2.7.5. Dados P y S, y siendo $n > 0$,

$$\min_{\lhd}(\Delta_S(P)) = s = i_1 \cdots i_n \;\Rightarrow\; \{\epsilon, i_1, i_1 i_2, \cdots, i_1 \cdots i_{n-1}\} \subseteq S$$

Demostración En primer lugar, observemos que $s \in \bigcap_{p \in P} \mathrm{Pos}(p)$ por definición de $\Delta_S(P)$. Sea $t = i_1 \cdots i_j$, con $0 \leq j < n$, un prefijo arbitrario de $i_1 \cdots i_n$. Debe suceder que $t \in \bigcap_{p \in P} \mathrm{Pos}(p)$ pues, en caso contrario, nunca podría ocurrir que $s \in \bigcap_{p \in P} \mathrm{Pos}(p)$. Además, para cualquier $p \in P$, ocurre que $p|_t \notin \mathcal{V}$ (si tal fuera el caso, s no sería una posición válida en p).

Supongamos ahora que $t \notin S$. El objetivo será concluir que $t \in \Delta_S(P)$, logrando así una contradicción dado que $t \lhd s$. Esta contradicción provendrá del hecho de suponer que $t \notin S$, habiendo probado así lo que deseábamos inicialmente.

Primero notemos que, si $|P| = 1$, se llegaría a una contradicción inmediata de acuerdo a la definición de $\Delta_S(P)$ para ese caso: $t \in \Delta_S(P)$. Por ende, asumamos que $|P| > 1$. Para probar que $t \in \Delta_S(P)$ bajo nuestras hipótesis, observemos que, según la definición, sólo resta demostrar que $|P'| = 1$ o bien $\Delta_{S \cup \{t\}}(P') \neq \emptyset$ para cualquier $P' \in P/_{\mathcal{R}_t}$. Si $|P'| > 1$, veremos que $\Delta_{S \cup \{t\}}(P') \neq \emptyset$, en particular, probaremos que $s \in \Delta_{S \cup \{t\}}(P')$. Para ello, veamos lo siguiente:

- $s \in \bigcap_{p \in P'} \text{Pos}(p)$ dado que $s \in \bigcap_{p \in P} \text{Pos}(p)$ y $P' \subseteq P$
- $s \notin S \cup \{t\}$ pues $s \notin S$ y t es un prefijo propio de s
- $\forall p \in P' : p|_s \notin \mathcal{V}$ pues $P' \subseteq P$ y dicha propiedad vale para P

Resta ver que, dado $Q \in P'/_{\mathcal{R}_s}$, $|Q| = 1$ o bien $\Delta_{S \cup \{t,s\}}(Q) \neq \emptyset$. En principio, notar que tal Q necesariamente pertenecerá a $P/_{\mathcal{R}_s}$ (como consecuencia del Lema 2.7.6) y, por ende, $|\Delta_{S \cup \{s\}}(Q)| > 0$ o bien $|Q| = 1$. De valer este último caso, habríamos conseguido lo buscado. Luego, supongamos que $\Delta_{S \cup \{s\}}(Q) \neq \emptyset$, y supongamos además que $\Delta_{S \cup \{s\}}(Q) = \{t\}$. Esto implica, por definición, que cada clase de equivalencia R de Q módulo $\mathcal{R}_t$ es tal que $|R| = 1$ o $\Delta_{S \cup \{s,t\}}(R) \neq \emptyset$.

Al ser t prefijo de s, y al ser Q una clase de equivalencia de P módulo $\mathcal{R}_s$, se tiene que $Q/_{\mathcal{R}_t} = \{Q\}$. En consecuencia, por lo dicho en el párrafo anterior, $|Q| = 1$ o bien $\Delta_{S \cup \{s,t\}}(Q) \neq \emptyset$. Pero esto es justamente a lo que queríamos llegar, con lo cual supongamos ahora que $t \notin \Delta_{S \cup \{s\}}(Q)$. Al ser tal conjunto no vacío, debe existir una posición $w \notin S \cup \{s\}$ tal que verifique lo requerido por la definición. Tal w, por la hipótesis que tomamos, es también distinta a t, con lo cual $w \in \Delta_{S \cup \{s,t\}}(Q)$. Así, logramos probar lo que buscábamos.

En conclusión, en cualquiera de los casos se llega a que $|Q| = 1$ o bien $\Delta_{S \cup \{t,s\}}(Q) \neq \emptyset$ para cualquier $Q \in P'/_{\mathcal{R}_s}$. Junto con los ítems previamente analizados, esto prueba que $s \in \Delta_{S \cup \{t\}}(P')$.

□

Este Lema sólo se utiliza como auxiliar del Lema 2.7.5. Prueba que, dado un conjunto de patrones P, y considerando que s es prefijo de t, cualquier clase de equivalencia de P módulo $\mathcal{R}_s$ particionada según $\mathcal{R}_t$ resulta en clases de equivalencia contenidas en $P/_{\mathcal{R}_t}$.

Lema 2.7.6. Sea $P \subset \mathcal{P}$, $s, t \in \bigcap_{p \in P} \text{Pos}(p)$ tales que $t = s \cdot w$ (i.e., s es prefijo de t), y $Q \in P/_{\mathcal{R}_s}$. Luego,

$$Q/_{\mathcal{R}_t} \subseteq P/_{\mathcal{R}_t}$$

Demostración En primera instancia, sean los patrones $p, q \in Q$. Si $(p, q) \in \mathcal{R}_t$, necesariamente ocurrirá que p y q estarán en la misma clase de equivalencia de P cocientado con $\mathcal{R}_t$.

Lo que resta ver es que, al cocientar a P inicialmente con $\mathcal{R}_s$, no puede suceder que dos patrones cualesquiera en diferentes clases de equivalencia se relacionen mediante $\mathcal{R}_t$. Es decir, dados $p, q \in P$, queremos ver que $(p, q) \notin \mathcal{R}_s \Rightarrow (p, q) \notin \mathcal{R}_t$.

Sean entonces tales p y q. Si $t = \epsilon$, entonces $s = \epsilon$, de lo cual trivialmente se obtiene que $(p, q) \notin \mathcal{R}_t$. Supongamos ahora que $t = t' \cdot i$, $i \in \mathbb{N}$, y que $(p, q) \in \mathcal{R}_t$. Luego, por definición,

$$\boldsymbol{c}(p|_t) = \boldsymbol{c}(q|_t) \wedge p \, \mathcal{R}_{t'} \, q$$

Ahora bien, $s = t'$ o bien s es prefijo propio de t'. En cualquier caso, razonando inductivamente se concluye que $(p, q) \notin \mathcal{R}_{t'}$ lo cual implica que $(p, q) \notin \mathcal{R}_t$. □

El Lema que sigue relaciona el tamaño de un patrón p ground cualquiera con la suma de las aridades de los constructores que aparecen en él.

Lema 2.7.7. Sea p un patrón ground. Luego,

$$\sum_{s \in \text{Pos}(p)} \delta(\boldsymbol{c}(p|_s)) = |p| - 1$$

Demostración Por inducción en p.

- $p = c \in \mathcal{C}$

$$\begin{aligned}\sum_{s \in \mathrm{Pos}(p)} \delta(\boldsymbol{c}(p|_s)) &= \delta(\boldsymbol{c}(p|_\epsilon)) \\ &= \delta(c) \\ &= 0 \\ &= |p| - 1\end{aligned}$$

- $p = c(p_1, \cdots, p_n)$, con $c \in \mathcal{C}$ de aridad n

$$\begin{aligned}\sum_{s \in \mathrm{Pos}(p)} \delta(\boldsymbol{c}(p|_s)) &= \delta(c) + \sum_{s \in \mathrm{Pos}(p) \setminus \{\epsilon\}} \delta(\boldsymbol{c}(p|_s)) \\ &= n + \sum_{i=1}^{n} \sum_{s \in \mathrm{Pos}(p_i)} \delta(\boldsymbol{c}(p_i|_s)) \\ &\overset{\text{HI}}{=} n + \sum_{i=1}^{n} (|p_i| - 1) \\ &= n + \sum_{i=1}^{n} |p_i| - n \\ &= |p| - 1\end{aligned}$$

□

El próximo Lema se utiliza únicamente en la Proposición 2.5.1 para el caso de reducción en la raíz. Por este motivo, utilizaremos aquí la misma notación que allí se utiliza. En esta Proposición, se tiene un término M donde una abstracción múltiple se aplica a cierta instancia de uno de sus patrones, donde dicha instancia se obtiene a través de una $\lambda\mathcal{C}$-sustitución σ:

$$M = (\lambda p_1.M_1 \mid \cdots \mid \lambda p_k.M_k)\ p_j^\sigma$$

Esta sustitución, entonces, opera sobre las variables que p_j liga en M_j (i.e., $\mathrm{Dom}(\sigma) \subseteq \mathrm{FV}(p_j)$).

Para los fines de la Proposición, interesa probar que traducir el cuerpo de la abstracción (M_j) con las variables instanciadas según σ es exactamente igual a aplicar cierta $\lambda\mathcal{B}_c$-sustitución τ_j al término que trae la sustitución ρ y sustituye la variable $\square_{p_j}$ introducida por Φ (i.e., $\Psi(M_j^{\omega_j})$, siguiendo lo expresado por Ψ-Abs). Esta sustitución τ_j tiene como dominio el conjunto de las variables en $\mathfrak{V}$ introducidas por ω_j junto con las variables libres de p_j, y mapea cada variable v a la traducción del término que σ asocia a la variable en p_j correspondiente.

Intuitivamente, el resultado de este Lema asegura que traducir un cuerpo de abstracción ya instanciado es lo mismo que traducir en primera instancia el esqueleto y luego sustituir allí las variables por las respectivas traducciones de sus mapeos.

Lema 2.7.8. Sea $M \in \mathfrak{M} \,/\, M = \lambda p_1.M_1 \mid \cdots \mid \lambda p_k.M_k$, donde cada $p_i \notin \mathcal{V}$, y sea σ una $\lambda\mathcal{C}$-sustitución tal que $\mathrm{Dom}(\sigma) \subseteq \mathrm{FV}(p_j)$, para algún j tal que $1 \leq j \leq k$. Entonces:

$$\begin{array}{lcl} \Psi(p_j^\sigma) & = & \Psi(p_j)^{\tau_j} \\ \Psi(M_j^\sigma) & = & \Psi(M_j^{\omega_j})^{\tau_j} \end{array}$$

en donde τ_j es la siguiente $\lambda\mathcal{B}_c$-sustitución:

$$\tau_j(v) = \begin{cases} \Psi(v^{\sigma \circ \omega_j^{-1}}) & \text{si } \exists x \in \mathrm{FV}(p_j) \,/\, \omega_j(x) = v \\ \Psi(v^\sigma) & \text{si } v \in \mathrm{FV}(p_j) \\ v & \text{en otro caso} \end{cases}$$

Demostración En principio, observar que τ_j está bien definida: si cierta variable x aparece en p_j, al estar entonces ligada en M, puede asumirse que $x \notin \mathfrak{V}$ por la convención de Barendregt.

La primera parte la probamos por inducción en p_j.

Respecto del caso base, notemos que p_j puede únicamente ser un constructor, y no una variable. De esta manera, el resultado sigue trivialmente pues las sustituciones (y la función Ψ) no tienen efecto sobre los constructores. Si, en cambio, $p_j = c(t_1, \cdots, t_n)$ para algún $n > 0$, podemos hacer la siguiente observación: si t_i es variable, para algún i, por supuesto se tiene que $t_i \in \mathrm{FV}(p_j)$, de manera que el resultado para t_i es inmediato a través de la definición de τ_j y de Ψ-Var. Entonces, a partir de esto,

$$\begin{array}{rcl} \Psi(p_j^\sigma) & = & \Psi(c(t_1, \cdots, t_n)^\sigma) \\ & \overset{\text{Sub-Pat-C}}{=} & \Psi(c(t_1^\sigma, \cdots, t_n^\sigma)) \\ & \overset{\Psi\text{-Pat}}{=} & c\ \Psi(t_1^\sigma) \cdots \Psi(t_n^\sigma) \\ & \overset{\text{HI + obs}}{=} & c\ \Psi(t_1)^{\tau_j} \cdots \Psi(t_n)^{\tau_j} \\ & \overset{\text{Sub-App-Bc}}{=} & (c\ \Psi(t_1) \cdots \Psi(t_n))^{\tau_j} \\ & \overset{\Psi\text{-Pat}}{=} & \Psi(c(t_1, \cdots, t_n))^{\tau_j} \\ & = & \Psi(p_j)^{\tau_j} \end{array}$$

Para la segunda parte procederemos, análogamente, por inducción en M_j.

- $M_j = x \in \mathcal{V}$

Supongamos que $x \in \mathrm{FV}(p_j)$. Sea entonces $v \in \mathfrak{V}$ tal que $\omega_j(x) = v$. Luego,

$$\begin{array}{rcl} \Psi(M_j^\sigma) & = & \Psi(x^\sigma) \\ & = & \Psi(v^{\sigma \circ \omega_j^{-1}}) \\ & = & v^{\tau_j} \\ & \overset{\Psi\text{-Var}}{=} & \Psi(v)^{\tau_j} \\ & = & \Psi(x^{\omega_j})^{\tau_j} \\ & = & \Psi(M_j^{\omega_j})^{\tau_j} \end{array}$$

Por otro lado, si $x \notin \mathrm{FV}(p_j)$, la sustitución σ no afectará a x. Además, $x \notin \mathfrak{V}$ pues estamos asumiendo que los términos, inicialmente, sólo poseen variables de $\mathcal{V}$. Por ende,

$x \notin \mathrm{Dom}(\tau_j)$ y, entonces,

$$\begin{array}{rcl}
\Psi(M_j^\sigma) & = & \Psi(x^\sigma) \\
& = & \Psi(x) \\
& \overset{\Psi\text{-Var}}{=} & x \\
& = & x^{\tau_j} \\
& \overset{\Psi\text{-Var}}{=} & \Psi(x)^{\tau_j} \\
& = & \Psi(x^{\omega_j})^{\tau_j} \\
& = & \Psi(M_j^{\omega_j})^{\tau_j}
\end{array}$$

- $M_j = c(N_1, \cdots, N_n)$, $c \in \mathcal{C}$

$$\begin{array}{rcl}
\Psi(M_j^\sigma) & = & \Psi(c(N_1, \cdots, N_n)^\sigma) \\
& \overset{\text{Sub-Pat-C}}{=} & \Psi(c(N_1^\sigma, \cdots, N_n^\sigma)) \\
& \overset{\Psi\text{-Pat}}{=} & c\ \Psi(N_1^\sigma) \cdots \Psi(N_n^\sigma) \\
& \overset{\text{HI}}{=} & c\ \Psi(N_1^{\omega_j})^{\tau_j} \cdots \Psi(N_n^{\omega_j})^{\tau_j} \\
& \overset{\text{Sub-App-Bc}}{=} & \left(c\ \Psi(N_1^{\omega_j}) \cdots \Psi(N_n^{\omega_j})\right)^{\tau_j} \\
& \overset{\Psi\text{-Pat}}{=} & \Psi(c(N_1^{\omega_j}, \cdots, N_n^{\omega_j}))^{\tau_j} \\
& \overset{\text{Sub-Pat-C}}{=} & \Psi(c(N_1, \cdots, N_n)^{\omega_j})^{\tau_j} \\
& = & \Psi(M_j^{\omega_j})^{\tau_j}
\end{array}$$

- $M_j = N_1\, N_2$

$$\begin{array}{rcl}
\Psi(M_j^\sigma) & = & \Psi((N_1\, N_2)^\sigma) \\
& \overset{\text{Sub-App-C}}{=} & \Psi(N_1^\sigma\ N_2^\sigma) \\
& \overset{\Psi\text{-App}}{=} & \Psi(N_1^\sigma)\, \Psi(N_2^\sigma) \\
& \overset{\text{HI}}{=} & \Psi(N_1^{\omega_j})^{\tau_j}\ \Psi(N_2^{\omega_j})^{\tau_j} \\
& \overset{\text{Sub-App-Bc}}{=} & \left(\Psi(N_1^{\omega_j})\, \Psi(N_2^{\omega_j})\right)^{\tau_j} \\
& \overset{\Psi\text{-App}}{=} & \Psi(N_1^{\omega_j}\ N_2^{\omega_j})^{\tau_j} \\
& \overset{\text{Sub-App-C}}{=} & \Psi((N_1\, N_2)^{\omega_j})^{\tau_j} \\
& = & \Psi(M_j^{\omega_j})^{\tau_j}
\end{array}$$

- $M_j = \lambda q_1.N_1 \mid \cdots \mid \lambda q_n.N_n$

En este caso,

$$\begin{array}{rcl}
\Psi(M_j^\sigma) & = & \Psi((\lambda q_1.N_1 \mid \cdots \mid \lambda q_n.N_n)^\sigma) \\
& \overset{\text{Sub-Abs-C}}{=} & \Psi(\lambda q_1.N_1^\sigma \mid \cdots \mid \lambda q_n.N_n^\sigma) \\
& \overset{\Psi\text{-Abs}}{=} & \lambda u.\Phi_u(\{q_1, \cdots, q_n\}, \emptyset)^\rho
\end{array}$$

con $\rho = \left\{ \Box_{q_1} \mapsto \Psi\left((N_1^\sigma)^{\omega_1'}\right) ; \cdots ; \Box_{q_n} \mapsto \Psi\left((N_n^\sigma)^{\omega_n'}\right) \right\}$. Para $1 \leq l \leq n$, por ser cada ω_l' un renombramiento de variables, será lo mismo aplicar dicha sustitución antes de traducir que después de traducir. Entonces:

$$\Psi\left((N_l^\sigma)^{\omega_l'}\right) = \Psi\,(N_l^\sigma)^{\omega_l'} \overset{\text{HI}}{=} \left(\Psi\left(N_l^{\omega_j}\right)^{\tau_j}\right)^{\omega_l'}$$

Ahora bien, $\mathrm{Dom}(\omega'_l) = \mathrm{FV}(q_l)$. En base a esto, tenemos lo siguiente:

- $\mathrm{Dom}(\omega'_l) \cap \mathrm{Dom}(\tau_j) = \emptyset$
  Por la convención de variables de Barendregt, cada variable en $\mathrm{Dom}(\omega'_l)$ puede asumirse distinta a las restantes pues éstas son las que q_l liga en N_l.
- $\mathrm{Dom}(\omega'_l) \cap \mathrm{VRan}(\tau_j) = \emptyset$
  Ídem.
- $\mathrm{Dom}(\tau_j) \cap \mathrm{VRan}(\omega'_l) = \emptyset$
  Cada variable presente en $\mathrm{VRan}(\omega'_l)$ es de la forma u_s. Dado que Ψ-Abs introduce una variable fresca, se tiene que $u \neq v$ para cualquier otra variable v.

De esta manera, se satisfacen las hipótesis del Lema 2.7.13, de manera que

$$\left(\Psi\left(N_l^{\omega_j}\right)^{\tau_j}\right)^{\omega'_l} = \left(\Psi\left(N_l^{\omega_j}\right)^{\omega'_l}\right)^{\tau_j} \quad (1 \leq l \leq n)$$

En consecuencia,

$$\rho = \left\{ \Box_{q_1} \mapsto \left(\Psi\left(N_1^{\omega_j}\right)^{\omega'_1}\right)^{\tau_j} ; \cdots ; \Box_{q_n} \mapsto \left(\Psi\left(N_n^{\omega_j}\right)^{\omega'_n}\right)^{\tau_j} \right\}$$

Esto permite decir lo siguiente:

$$\lambda u.\Phi_u(\{q_1, \cdots, q_n\}, \emptyset)^{\rho} = \left(\lambda u.\Phi_u(\{q_1, \cdots, q_n\}, \emptyset)^{\rho'}\right)^{\tau_j}$$

siendo $\rho' = \{ \Box_{q_1} \mapsto \Psi\left((N_1^{\omega_j})^{\omega'_1}\right) ; \cdots ; \Box_{q_n} \mapsto \Psi\left((N_n^{\omega_j})^{\omega'_n}\right) \}$. Esta igualdad se justifica por construcción de $\Phi_u(\{q_1, \cdots, q_n\}, \emptyset)^{\rho'}$; las únicas variables libres que aparecerán en este término, como consecuencia del Corolario 2.7.4, son aquellas presentes en cada $\Psi\left((N_l^{\omega_j})^{\omega'_l}\right)$, además de u. No obstante, este hecho no influye en el resultado pues τ_j no tiene efecto sobre tal variable.

Finalmente, notemos que $\lambda u.\Phi_u(\{q_1, \cdots, q_n\}, \emptyset)^{\rho'} = \Psi(M_j^{\omega_j}) \Rightarrow \Psi(M_j^{\sigma}) = \Psi(M_j^{\omega_j})^{\tau_j}$

□

El siguiente Lema indica que la traducción Ψ es cerrada por sustituciones, lo que permite resolver en la simulación el caso de reducción clásica en la raíz.

Lema 2.7.9. Sean $M, N \in \mathfrak{M}$ y $x \in \mathcal{V}$. Luego,

$$\Psi(M)[x \,/\, \Psi(N)] = \Psi(M[x \,/\, N])$$

Demostración Por inducción en M.

- $M = y \in \mathcal{V} \cup \mathfrak{V}$

Si $y \neq x$:

$$\Psi(y)[x \,/\, \Psi(N)] \overset{\Psi\text{-Var}}{=} y[x \,/\, \Psi(N)] \overset{\text{Sub-Var-Bc}}{=} y \overset{\Psi\text{-Var}}{=} \Psi(y) \overset{\text{Sub-Var-C}}{=} \Psi(y[x \,/\, N])$$

Por otro lado, si $y = x$:

$$\Psi(x)[x \,/\, \Psi(N)] \overset{\Psi\text{-Var}}{=} x[x \,/\, \Psi(N)] \overset{\text{Sub-Var-Bc}}{=} \Psi(N) \overset{\text{Sub-Var-C}}{=} \Psi(x[x \,/\, N])$$

- $M = c(M_1, \cdots, M_n), c \in \mathcal{C}$

$$
\begin{array}{rcl}
\Psi(M)[x \,/\, \Psi(N)] & = & \Psi\left(c(M_1, \cdots, M_n)\right)[x \,/\, \Psi(N)] \\
& \overset{\Psi\text{-Pat}}{=} & \left(c\,\Psi(M_1) \cdots \Psi(M_n)\right)[x \,/\, \Psi(N)] \\
& \overset{\text{Sub-App-Bc}}{=} & c\,\Psi(M_1)[x \,/\, \Psi(N)] \cdots \Psi(M_n)[x \,/\, \Psi(N)] \\
& \overset{\text{HI}}{=} & c\,\Psi(M_1[x \,/\, N]) \cdots \Psi(M_n[x \,/\, N]) \\
& \overset{\Psi\text{-Pat}}{=} & \Psi\left(c(M_1[x \,/\, N], \cdots, M_n[x \,/\, N])\right) \\
& \overset{\text{Sub-Pat-C}}{=} & \Psi\left(c(M_1, \cdots, M_n)[x \,/\, N]\right) \\
& = & \Psi\left(M[x \,/\, N]\right)
\end{array}
$$

- $M = M_1\, M_2$

$$
\begin{array}{rcl}
\Psi(M)[x \,/\, \Psi(N)] & = & \Psi\left(M_1\, M_2\right)[x \,/\, \Psi(N)] \\
& \overset{\Psi\text{-App}}{=} & \left(\Psi(M_1)\,\Psi(M_2)\right)[x \,/\, \Psi(N)] \\
& \overset{\text{Sub-App-Bc}}{=} & \Psi(M_1)[x \,/\, \Psi(N)]\,\Psi(M_2)[x \,/\, \Psi(N)] \\
& \overset{\text{HI}}{=} & \Psi(M_1[x \,/\, N])\,\Psi(M_2[x \,/\, N]) \\
& \overset{\Psi\text{-App}}{=} & \Psi\left(M_1[x \,/\, N]\, M_2[x \,/\, N]\right)) \\
& \overset{\text{Sub-App-C}}{=} & \Psi\left((M_1\, M_2)[x \,/\, N]\right) \\
& = & \Psi\left(M[x \,/\, N]\right)
\end{array}
$$

- $M = \lambda p_1.M_1 | \cdots | \lambda p_k.M_k$

$$
\begin{array}{rcl}
\Psi(M)[x \,/\, \Psi(N)] & = & \Psi\left(\lambda p_1.M_1 | \cdots | \lambda p_k.M_k\right)[x \,/\, \Psi(N)] \\
& \overset{\Psi\text{-Abs}}{=} & \left(\lambda v.\Phi_v(\{p_1, \cdots, p_k\}, \emptyset)^\rho\right)[x \,/\, \Psi(N)] \\
& \overset{\text{Sub-Abs-Bc}}{=} & \lambda v.\Phi_v(\{p_1, \cdots, p_k\}, \emptyset)^\rho[x \,/\, \Psi(N)] \\
& = & \lambda v.\Phi_v(\{p_1, \cdots, p_k\}, \emptyset)^{\{x \mapsto \Psi(N)\} \circ \rho} \\
& = & \lambda v.\Phi_v(\{p_1, \cdots, p_k\}, \emptyset)^{\left\{\, \square_{p_1} \mapsto \Psi(M_1^{\omega_1})[x \,/\, \Psi(N)], \cdots, \square_{p_k} \mapsto \Psi(M_k^{\omega_k})[x \,/\, \Psi(N)], x \mapsto \Psi(N)\,\right\}} \\
& \overset{(1)}{=} & \lambda v.\Phi_v(\{p_1, \cdots, p_k\}, \emptyset)^{\left\{\, \square_{p_1} \mapsto \Psi(M_1^{\omega_1})[x \,/\, \Psi(N)], \cdots, \square_{p_k} \mapsto \Psi(M_k^{\omega_k})[x \,/\, \Psi(N)]\,\right\}} \\
& \overset{(2)}{=} & \lambda v.\Phi_v(\{p_1, \cdots, p_k\}, \emptyset)^{\left\{\, \square_{p_1} \mapsto (\Psi(M_1)[x \,/\, \Psi(N)])^{\omega_1}, \cdots, \square_{p_k} \mapsto (\Psi(M_k)[x \,/\, \Psi(N)])^{\omega_k}\,\right\}} \\
& \overset{\text{HI}}{=} & \lambda v.\Phi_v(\{p_1, \cdots, p_k\}, \emptyset)^{\left\{\, \square_{p_1} \mapsto \Psi(M_1[x \,/\, N])^{\omega_1}, \cdots, \square_{p_k} \mapsto \Psi(M_k[x \,/\, N])^{\omega_k}\,\right\}} \\
& \overset{(3)}{=} & \lambda v.\Phi_v(\{p_1, \cdots, p_k\}, \emptyset)^{\left\{\, \square_{p_1} \mapsto \Psi(M_1[x \,/\, N]^{\omega_1}), \cdots, \square_{p_k} \mapsto \Psi(M_k[x \,/\, N]^{\omega_k})\,\right\}} \\
& \overset{(4)}{=} & \Psi\left(\lambda p_1.M_1[x \,/\, N] \,|\, \cdots \,|\, \lambda p_k.M_k[x \,/\, N]\right) \\
& \overset{\text{Sub-Abs-C}}{=} & \Psi\left((\lambda p_1.M_1 | \cdots | \lambda p_k.M_k)[x \,/\, N]\right) \\
& = & \Psi\left(M[x \,/\, N]\right)
\end{array}
$$

La igualdad (1) es consecuencia del Lema 2.7.3: $x \notin \mathrm{FV}(\Phi_v(\{p_1, \cdots, p_k\}, \emptyset))$.

Dado que cada ω_j es un renombramiento de variables, es indistinto aplicarlos antes o después de traducir. Esto justifica las igualdades (2) y (3), notando además que en (2) se utiliza el Lema 2.7.13, cuyas hipótesis son ciertas por la convención de Barendregt.

Por último, observar en (4) que el renombramiento ω_j de cada término M_j es equivalente al del término $M_j[x \,/\, N]$: las únicas variables ligadas de estos últimos términos no serán otras que aquellas ligadas en el respectivo término original.

□

Los siguientes dos lemas se utilizan para acotar el tamaño de la traducción, en el caso de los términos generados a través de Φ. La cota, en estos casos, se da en función del tamaño de los patrones de P.

Lema 2.7.10. Sea $(P, S) \in \mathrm{Dom}(\Phi)$, siendo $P = \{p_1, \cdots, p_k\}$, y sea $v \in \mathfrak{V}$. Entonces,

$$|\Phi_v(P, S)| \leq \sum_{i=1}^{k} |\Phi_v(\{p_i\}, S)|$$

Demostración Por inducción en $|\bigcup_{p \in P} \mathrm{Pos}(p) \setminus S|$.

Si dicho valor es cero, la Observación 2.4.1 asegura que $|P| = 1$, i.e., $k = 1$. De esta manera, el resultado es trivial. De lo contrario, es preciso analizar los siguientes escenarios:

- $P = \{p\} \wedge p|_s \in \mathcal{V}$

Dado que $k = 1$, el resultado es trivialmente cierto.

- $P = \{p\} \wedge p|_s \notin \mathcal{V}$

Ídem caso anterior.

- $k > 1$

Sean

- $s = \min_{\lhd}(\Delta_S(P))$
- Para cada $i = 1, \cdots, m$, $P_i = \{p_1^i, \cdots, p_{r_i}^i\}$, donde $\{P_1, \cdots, P_m\} = P \diagup \mathcal{R}_s$

Luego,

$$
\begin{aligned}
|\Phi_v(\{p_1,\cdots,p_k\},S)| &\overset{\Phi\text{-}2}{=} |\{\!|c_1 \mapsto T_1;\cdots;c_m \mapsto T_m|\!\}.v_s| \\
&\overset{\text{S-Case-Bc}}{=} 3+2m+\sum_{i=1}^{m}|T_i| \\
&= 3+2m+\sum_{i=1}^{m}|\lambda v_{s\cdot 1}\cdots v_{s\cdot\delta(c_i)}.\Phi_v(P_i,S\cup\{s\})| \\
&\overset{\text{S-Abs-Bc}}{=} 3+2m+\sum_{i=1}^{m}(2\delta(c_i)+|\Phi_v(P_i,S\cup\{s\})|) \\
&\overset{\text{HI}}{\leq} 3+2m+\sum_{i=1}^{m}\left(2\delta(c_i)+\sum_{j=1}^{r_i}|\Phi_v(\{p_j^i\},S\cup\{s\})|\right) \\
&\overset{r_i\geq 1}{\leq} 3+2m+\sum_{i=1}^{m}\left(2r_i\delta(c_i)+\sum_{j=1}^{r_i}|\Phi_v(\{p_j^i\},S\cup\{s\})|\right) \\
&= 3+2m+\sum_{i=1}^{m}\sum_{j=1}^{r_i}\left(2\delta(c_i)+|\Phi_v(\{p_j^i\},S\cup\{s\})|\right) \\
&\overset{\text{S-Abs-Bc}}{=} 3+2m+\sum_{i=1}^{m}\sum_{j=1}^{r_i}|\lambda v_{s\cdot 1}\cdots v_{s\cdot\delta(c_i)}.\Phi_v(\{p_j^i\},S\cup\{s\})| \\
&\leq 3+2k+\sum_{i=1}^{m}\sum_{j=1}^{r_i}|\lambda v_{s\cdot 1}\cdots v_{s\cdot\delta(c_i)}.\Phi_v(\{p_j^i\},S\cup\{s\})| \\
&\leq 5k+\sum_{i=1}^{m}\sum_{j=1}^{r_i}|\lambda v_{s\cdot 1}\cdots v_{s\cdot\delta(c_i)}.\Phi_v(\{p_j^i\},S\cup\{s\})| \\
&= \sum_{i=1}^{m}\sum_{j=1}^{r_i}5+|\lambda v_{s\cdot 1}\cdots v_{s\cdot\delta(c_i)}.\Phi_v(\{p_j^i\},S\cup\{s\})| \\
&\overset{\text{S-Case-Bc}}{=} \sum_{i=1}^{m}\sum_{j=1}^{r_i}|\{\!|\boldsymbol{c}(p_j^i|_s)\mapsto\lambda v_{s\cdot 1}\cdots v_{s\cdot\delta(c_i)}.\Phi_v(\{p_j^i\},S\cup\{s\})|\!\}.v_s| \\
&= \sum_{i=1}^{m}\sum_{j=1}^{r_i}|\Phi_v(\{p_j^i\},S)| \\
&= \sum_{i=1}^{k}|\Phi_v(\{p_i\},S)|
\end{aligned}
$$

Notar que, en la anteúltima igualdad, podría suceder que $s \neq \min_{\lhd}(\Delta_S(\{p_j^i\}))$. No obstante, la igualdad se mantiene dado que los tamaños serán exactamente iguales; lo que cambia es la forma de construir el término, explorando las posiciones en un orden distinto según el caso. □

Lema 2.7.11. Sea $(\{p\},S)\in\mathrm{Dom}(\Phi)$ y sea $v\in\mathfrak{V}$. Luego,

$$|\Phi_v(\{p\},S)| \leq 7\,|p|-1$$

Demostración Observando la ecuación Φ-1, se deduce que el tamaño de $\Phi_v(\{p\}, S)$ se maximiza si cada posición $w \in \mathrm{Pos}(p) \setminus S$ es tal que $p|_w \notin \mathcal{V}$. De esta forma, podemos acotar superiormente el tamaño de dicho término por el tamaño del término $\Phi_v(\{q\}, S)$, donde $q = p^\sigma$, siendo σ la sustitución que mapea cada variable de p a un mismo constructor $a \in \mathcal{C}$ tal que $\delta(a) = 0$ (por ende, $|p| = |q|$).

Sea $\{s_1, \cdots, s_k\} = \Delta_S(\{q\})$ de manera tal que $s_i \lhd s_j$ si $i < j$. Luego,

$$
\begin{array}{rcl}
|\Phi_v(\{p\}, S)| & \leq & |\Phi_v(\{q\}, S)| \\
& = & |\{\!|\boldsymbol{c}(q|_{s_1}) \mapsto \lambda v_{s_1 \cdot 1} \cdots v_{s_1 \cdot \delta(\boldsymbol{c}(q|_{s_1}))}.\Phi_v(\{q\}, S \cup \{s_1\})|\!\}.v_{s_1}| \\
& \overset{\text{S-Case-Bc, S-Abs-Bc}}{=} & (5 + 2\delta(\boldsymbol{c}(q|_{s_1}))) + |\Phi_v(\{q\}, S \cup \{s_1\})| \\
& = & (5 + 2\delta(\boldsymbol{c}(q|_{s_1}))) + \\
& & |\{\!|\boldsymbol{c}(q|_{s_2}) \mapsto \lambda v_{s_2 \cdot 1} \cdots v_{s_2 \cdot \delta(\boldsymbol{c}(q|_{s_2}))}.\Phi_v(\{q\}, S \cup \{s_1, s_2\})|\!\}.v_{s_2}| \\
& \overset{\text{S-Case-Bc, S-Abs-Bc}}{=} & (5 + 2\delta(\boldsymbol{c}(q|_{s_1}))) + (5 + 2\delta(\boldsymbol{c}(q|_{s_2}))) + |\Phi_v(\{q\}, S \cup \{s_1, s_2\})| \\
& \vdots & \\
& = & \displaystyle\sum_{s \in \mathrm{Pos}(q) \setminus S} (5 + 2\delta(\boldsymbol{c}(q|_s))) + |\Box_q| \\
& \overset{\text{S-Var-Bc}}{=} & \displaystyle\sum_{s \in \mathrm{Pos}(q) \setminus S} (5 + 2\delta(\boldsymbol{c}(q|_s))) + 1 \\
& \leq & \displaystyle\sum_{s \in \mathrm{Pos}(q)} (5 + 2\delta(\boldsymbol{c}(q|_s))) + 1 \\
& = & \displaystyle 5|q| + 2 \sum_{s \in \mathrm{Pos}(q)} \delta(\boldsymbol{c}(q|_s)) + 1 \\
& \overset{\text{Lema 2.7.7}}{=} & 5|q| + 2(|q| - 1) + 1 \\
& = & 7|q| - 1 \\
& = & 7|p| - 1
\end{array}
$$

□

El siguiente Lema indica cómo crece en tamaño un término al que se aplica una sustitución, conociendo los tamaños de su valor en las variables. Extiende de la manera esperada los casos del λ-cálculo y sistemas de reescritura de términos clásicos.

Lema 2.7.12. Sea $M \in \Lambda\mathcal{B}_c$, σ una sustitución y x una variable. Entonces,

$$|M^\sigma| = |M| + \sum_{x \,\in\, \mathrm{Dom}(\sigma)} |M|_x \; (|\sigma(x)| - 1)$$

Demostración Por inducción en M.

- $M = y \in \mathrm{Var}$

Supongamos que $y \in \mathrm{Dom}(\sigma)$. Luego,

$$\begin{array}{rcl} |M^\sigma| & = & |y^\sigma| \\ & = & 1 + |y^\sigma| - 1 \\ & \overset{\text{S-Var-Bc}}{=} & |y| + \sum_{x \in \mathrm{Dom}(\sigma)} |y|_x \ (|\sigma(x)| - 1) \\ & = & |M| + \sum_{x \in \mathrm{Dom}(\sigma)} |M|_x \ (|\sigma(x)| - 1) \end{array}$$

pues $|y|_x = 0$ si $x \neq y$.

Si $y \notin \mathrm{Dom}(\sigma)$, $|y|_x = 0 \ \forall x \in \mathrm{Dom}(\sigma) \Rightarrow \sum_{x \in \mathrm{Dom}(\sigma)} |M|_x \ (|\sigma(x)| - 1) = 0$. De esta manera, el lado derecho de la igualdad es exactamente $|y|$, y por ende la igualdad se satisface al tener que $|y^\sigma| = |y|$.

- $M = c \in \mathcal{C}$

La igualdad es trivialmente cierta dado que las sustituciones no tienen efecto sobre los constructores, y que además $|c|_x = 0 \ \forall x \in \mathrm{Var}$.

- $M = M_1 \, M_2$

$$\begin{array}{rcl} |M^\sigma| & = & |(M_1 \, M_2)^\sigma| \\ & \overset{\text{Sub-App-Bc}}{=} & |M_1^\sigma \, M_2^\sigma| \\ & \overset{\text{S-App-Bc}}{=} & 1 + |M_1^\sigma| + |M_2^\sigma| \\ & \overset{\text{HI}}{=} & 1 + |M_1| + \sum_{x \in \mathrm{Dom}(\sigma)} |M_1|_x \ (|\sigma(x)| - 1) + |M_2| + \sum_{x \in \mathrm{Dom}(\sigma)} |M_2|_x \ (|\sigma(x)| - 1) \\ & \overset{\text{S-App-Bc}}{=} & |M_1 \, M_2| + \sum_{x \in \mathrm{Dom}(\sigma)} (|M_1|_x + |M_2|_x) \ (|\sigma(x)| - 1) \\ & \overset{\#\text{-App}}{=} & |M_1 \, M_2| + \sum_{x \in \mathrm{Dom}(\sigma)} |M_1 \, M_2|_x \ (|\sigma(x)| - 1) \\ & = & |M| + \sum_{x \in \mathrm{Dom}(\sigma)} |M|_x \ (|\sigma(x)| - 1) \end{array}$$

- $M = \lambda z.M_1$

La variable z está ligada en M_1. Así, por la convención de Barendregt, vamos a asumir que

$z \notin \mathrm{Dom}(\sigma)$. Hecha esta observación,

$$\begin{array}{rcl}
|M^\sigma| & = & |(\lambda z.M_1)^\sigma| \\
& \overset{\text{Sub-Abs-Bc}}{=} & |\lambda z.M_1^\sigma| \\
& \overset{\text{S-Abs-Bc}}{=} & 2 + |M_1^\sigma| \\
& \overset{\text{HI}}{=} & 2 + |M_1| + \sum_{x \in \mathrm{Dom}(\sigma)} |M_1|_x \ (|\sigma(x)| - 1) \\
& \overset{\text{S-Abs-Bc}}{=} & |\lambda z.M_1| + \sum_{x \in \mathrm{Dom}(\sigma)} |M_1|_x \ (|\sigma(x)| - 1) \\
& \overset{\#\text{-Abs}}{=} & |\lambda z.M_1| + \sum_{x \in \mathrm{Dom}(\sigma)} |\lambda z.M_1|_x \ (|\sigma(x)| - 1) \\
& = & |M| + \sum_{x \in \mathrm{Dom}(\sigma)} |M|_x \ (|\sigma(x)| - 1)
\end{array}$$

- $M = \{\!|\, c_1 \mapsto M_1 \,;\, \cdots \,;\, c_k \mapsto M_k \,|\!\}.N$

$$\begin{array}{rcl}
|M^\sigma| & = & |\,(\{\!|\, c_1 \mapsto M_1 \,;\, \cdots \,;\, c_k \mapsto M_k \,|\!\}.N)^\sigma\,| \\
& \overset{\text{Sub-Case-Bc}}{=} & |\{\!|\, c_1 \mapsto M_1^\sigma \,;\, \cdots \,;\, c_k \mapsto M_k^\sigma \,|\!\}.N^\sigma| \\
& \overset{\text{S-Case-Bc}}{=} & 2 + 2k + |N^\sigma| + \sum_{i=1}^{k} |M_i^\sigma| \\
& \overset{\text{HI}}{=} & 2 + 2k + |N| + \sum_{x \in \mathrm{Dom}(\sigma)} |N|_x \ (|\sigma(x)| - 1) + \\
& & \sum_{i=1}^{k} \left(|M_i| + \sum_{x \in \mathrm{Dom}(\sigma)} |M_i|_x \ (|\sigma(x)| - 1) \right) \\
& = & 2 + 2k + |N| + \sum_{i=1}^{k} |M_i| + \sum_{x \in \mathrm{Dom}(\sigma)} |N|_x \ (|\sigma(x)| - 1) + \\
& & \sum_{i=1}^{k} \sum_{x \in \mathrm{Dom}(\sigma)} |M_i|_x \ (|\sigma(x)| - 1) \\
& \overset{\text{S-Case-Bc}}{=} & |\{\!|\, c_1 \mapsto M_1 \,;\, \cdots \,;\, c_k \mapsto M_k \,|\!\}.N| + \\
& & \sum_{x \in \mathrm{Dom}(\sigma)} \left(|N|_x + \sum_{i=1}^{k} |M_i|_x \right) (|\sigma(x)| - 1) \\
& \overset{\#\text{-Case}}{=} & |M| + \sum_{x \in \mathrm{Dom}(\sigma)} |\{\!|\, c_1 \mapsto M_1 \,;\, \cdots \,;\, c_k \mapsto M_k \,|\!\}.N|_x \ (|\sigma(x)| - 1) \\
& = & |M| + \sum_{x \in \mathrm{Dom}(\sigma)} |M|_x \ (|\sigma(x)| - 1)
\end{array}$$

□

El Lema que se muestra a continuación asegura la conmutatividad de la aplicación de sustituciones en $\lambda\mathcal{B}_c$ bajo ciertas hipótesis.

Lema 2.7.13. Sea $M \in \Lambda\mathcal{B}_c$ y sean σ_1, σ_2 sustituciones tales que:

- $\mathrm{Dom}(\sigma_1) \cap \mathrm{Dom}(\sigma_2) = \emptyset$
- $\mathrm{Dom}(\sigma_1) \cap \mathrm{VRan}(\sigma_2) = \emptyset$
- $\mathrm{Dom}(\sigma_2) \cap \mathrm{VRan}(\sigma_1) = \emptyset$

Entonces, $(M^{\sigma_1})^{\sigma_2} = (M^{\sigma_2})^{\sigma_1}$

Demostración Por inducción en M.

- $M = x \in \mathrm{Var}$

Supongamos que $x \in \mathrm{Dom}(\sigma_1) \Rightarrow x \notin \mathrm{Dom}(\sigma_2)$, i.e., $x^{\sigma_2} = x$. Además, si $y \in \mathrm{Dom}(\sigma_2) \Rightarrow y \notin \mathrm{VRan}(\sigma_1)$. De esto se deduce que $N^{\sigma_2} = N$ siempre que N esté en la imagen de σ_1 (i.e., $\exists z \in \mathrm{Var} : N = z^{\sigma_1}$). Entonces, si $x^{\sigma_1} = P$:

$$(M^{\sigma_1})^{\sigma_2} = (x^{\sigma_1})^{\sigma_2} = P^{\sigma_2} = P = x^{\sigma_1} = (x^{\sigma_2})^{\sigma_1} = (M^{\sigma_2})^{\sigma_1}$$

Por otro lado, si $x \notin \mathrm{Dom}(\sigma_1)$ pero $x \in \mathrm{Dom}(\sigma_2)$, el razonamiento es análogo al caso anterior. Si, en cambio, $x \notin \mathrm{Dom}(\sigma_2)$, ocurre que:

$$(M^{\sigma_1})^{\sigma_2} = (x^{\sigma_1})^{\sigma_2} = x^{\sigma_2} = x = x^{\sigma_1} = (x^{\sigma_2})^{\sigma_1} = (M^{\sigma_2})^{\sigma_1}$$

- $M = c \in \mathcal{C}$

Este caso es trivial pues las sustituciones no afectan a los constructores.

- $M = M_1\, M_2$

$$\begin{array}{rcl}
(M^{\sigma_1})^{\sigma_2} & = & ((M_1\, M_2)^{\sigma_1})^{\sigma_2} \\
& \overset{\text{Sub-App-Bc}}{=} & (M_1^{\sigma_1}\, M_2^{\sigma_1})^{\sigma_2} \\
& \overset{\text{Sub-App-Bc}}{=} & (M_1^{\sigma_1})^{\sigma_2}\, (M_2^{\sigma_1})^{\sigma_2} \\
& \overset{\text{HI}}{=} & (M_1^{\sigma_2})^{\sigma_1}\, (M_2^{\sigma_2})^{\sigma_1} \\
& \overset{\text{Sub-App-Bc}}{=} & (M_1^{\sigma_2}\, M_2^{\sigma_2})^{\sigma_1} \\
& \overset{\text{Sub-App-Bc}}{=} & ((M_1\, M_2)^{\sigma_2})^{\sigma_1} \\
& = & (M^{\sigma_2})^{\sigma_1}
\end{array}$$

- $M = \lambda x.M_1$

La variable x aparece ligada en M_1, por lo que, por la convención de Barendregt, asumimos que $x \notin \mathrm{Dom}(\sigma_1) \cup \mathrm{Dom}(\sigma_2)$. De esta manera,

$$\begin{array}{rcl}
(M^{\sigma_1})^{\sigma_2} & = & ((\lambda x.M_1)^{\sigma_1})^{\sigma_2} \\
& \overset{\text{Sub-Abs-Bc}}{=} & (\lambda x.M_1^{\sigma_1})^{\sigma_2} \\
& \overset{\text{Sub-Abs-Bc}}{=} & \lambda x.(M_1^{\sigma_1})^{\sigma_2} \\
& \overset{\text{HI}}{=} & \lambda x.(M_1^{\sigma_2})^{\sigma_1} \\
& \overset{\text{Sub-Abs-Bc}}{=} & (\lambda x.M_1^{\sigma_2})^{\sigma_1} \\
& \overset{\text{Sub-Abs-Bc}}{=} & ((\lambda x.M_1)^{\sigma_2})^{\sigma_1} \\
& = & (M^{\sigma_2})^{\sigma_1}
\end{array}$$

- $M = \{\!|\, c_1 \mapsto M_1 \,;\, \cdots \,;\, c_k \mapsto M_k \,|\!\}.N$

$$\begin{array}{rcl}
(M^{\sigma_1})^{\sigma_2} & = & ((\{\!|\, c_1 \mapsto M_1 \,;\, \cdots \,;\, c_k \mapsto M_k \,|\!\}.N)^{\sigma_1})^{\sigma_2} \\
& \overset{\text{Sub-Case-Bc}}{=} & \left(\{\!|\, c_1 \mapsto M_1^{\sigma_1} \,;\, \cdots \,;\, c_k \mapsto M_k^{\sigma_1} \,|\!\}.N^{\sigma_1}\right)^{\sigma_2} \\
& \overset{\text{Sub-Case-Bc}}{=} & \{\!|\, c_1 \mapsto (M_1^{\sigma_1})^{\sigma_2} \,;\, \cdots \,;\, c_k \mapsto (M_k^{\sigma_1})^{\sigma_2} \,|\!\}.(N^{\sigma_1})^{\sigma_2} \\
& \overset{\text{HI}}{=} & \{\!|\, c_1 \mapsto (M_1^{\sigma_2})^{\sigma_1} \,;\, \cdots \,;\, c_k \mapsto (M_k^{\sigma_2})^{\sigma_1} \,|\!\}.(N^{\sigma_2})^{\sigma_1} \\
& \overset{\text{Sub-Case-Bc}}{=} & \left(\{\!|\, c_1 \mapsto M_1^{\sigma_2} \,;\, \cdots \,;\, c_k \mapsto M_k^{\sigma_2} \,|\!\}.N^{\sigma_2}\right)^{\sigma_1} \\
& \overset{\text{Sub-Case-Bc}}{=} & ((\{\!|\, c_1 \mapsto M_1 \,;\, \cdots \,;\, c_k \mapsto M_k \,|\!\}.N)^{\sigma_2})^{\sigma_1} \\
& = & (M^{\sigma_2})^{\sigma_1}
\end{array}$$

□

Los dos lemas que siguen, que tratan sobre preservación de la reducción en $\lambda\mathcal{B}_c$ bajo ciertas condiciones, se utilizan en forma auxiliar a la Proposición 2.5.1.

Lema 2.7.14. Sean $M, N, P \in \Lambda\mathcal{B}_c$ y sea σ una sustitución tal que $\mathrm{Dom}(\sigma) \cap \mathrm{FV}(M) = \emptyset$. Entonces,

$$M[x \,/\, P] \to_{\lambda\mathcal{B}_c} N \;\Rightarrow\; M[x \,/\, P^{\sigma}] \to_{\lambda\mathcal{B}_c} N^{\sigma}$$

Demostración Sea $P \in \Lambda\mathcal{B}_c$. Si $x \notin \mathrm{FV}(M) \;\Rightarrow\; M[x \,/\, P] = M \to_{\lambda\mathcal{B}_c} N$. Dado que $\mathrm{Dom}(\sigma) \cap \mathrm{FV}(M) = \emptyset$ y, por reducir M a N, $\mathrm{FV}(N) \subseteq \mathrm{FV}(M)$, se tiene que $\mathrm{Dom}(\sigma) \cap \mathrm{FV}(N) = \emptyset$, con lo cual $N^{\sigma} = N \;\Rightarrow\; M[x \,/\, P^{\sigma}] = M \to_{\lambda\mathcal{B}_c} N = N^{\sigma}$.

Por otro lado, si $x \in \mathrm{FV}(M) \;\Rightarrow\; x \notin \mathrm{Dom}(\sigma)$, y tenemos que:

$$M[x \,/\, P^{\sigma}] \overset{\text{Hip.}}{=} M^{\sigma}[x \,/\, P^{\sigma}] = (M[x \,/\, P])^{\sigma}$$

donde la última igualdad es consecuencia del Lema 2.7.16, por las hipótesis $\mathrm{Dom}(\sigma) \cap \mathrm{FV}(M) = \emptyset$ y $x \notin \mathrm{Dom}(\sigma)$. Finalmente, valiéndonos de que $\to_{\lambda\mathcal{B}_c}$ es cerrada por sustituciones [5] y de que $M[x \,/\, P] \to_{\lambda\mathcal{B}_c} N$, obtenemos lo que deseábamos demostrar.

□

Corolario 2.7.15. Sean $M, N, P \in \Lambda\mathcal{B}_c$ y sea σ una sustitución tal que $\mathrm{Dom}(\sigma) \cap \mathrm{FV}(M) = \emptyset$. Entonces,

$$M[x \,/\, P] \overset{n}{\twoheadrightarrow}_{\lambda\mathcal{B}_c} N \;\Rightarrow\; M[x \,/\, P^{\sigma}] \overset{n}{\twoheadrightarrow}_{\lambda\mathcal{B}_c} N^{\sigma}$$

Demostración Para $n = 0$, el resultado se obtiene siguiendo un razonamiento similar al realizado para el Lema previo. En otro caso,

$$M[x \,/\, P] \to N' \overset{n-1}{\twoheadrightarrow}_{\lambda\mathcal{B}_c} N$$

El resultado sigue a partir del Lema anterior y usando que $\twoheadrightarrow_{\lambda\mathcal{B}_c}$ es cerrada por sustituciones [5].

□

El próximo Lema generaliza, de forma natural, no sólo el Lema de Sustitución clásico sino también el mismo de $\lambda\mathcal{B}_c$.

Lema 2.7.16. Sean $M, P \in \Lambda\mathcal{B}_c$, $x \in \mathrm{Var}$ y σ una sustitución tal que:

1. $x \notin \mathrm{Dom}(\sigma)$
2. $x \notin \mathrm{VRan}(\sigma) \vee \mathrm{Dom}(\sigma) \cap \mathrm{FV}(M) = \emptyset$.

Luego, $(M[x \,/\, P])^\sigma = M^\sigma[x \,/\, P^\sigma]$.

Demostración Por inducción en M.

- $M = y \in \mathrm{Var}$

  Supongamos que $y = x$. Entonces:

$$M^\sigma[x \,/\, P^\sigma] = x^\sigma[x \,/\, P^\sigma] \overset{1.}{=} x[x \,/\, P^\sigma] = P^\sigma = (x[x \,/\, P])^\sigma = (M[x \,/\, P])^\sigma$$

Por otra parte, si $y \neq x$, supongamos que $x \notin \mathrm{VRan}(\sigma)$, i.e., $x \notin \mathrm{FV}(z^\sigma)\ \forall z \in \mathrm{Dom}(\sigma)$. Luego,

$$M^\sigma[x \,/\, P^\sigma] = y^\sigma[x \,/\, P^\sigma] = y^\sigma = (y[x \,/\, P])^\sigma = (M[x \,/\, P])^\sigma$$

Si $\mathrm{Dom}(\sigma) \cap \mathrm{FV}(M) = \emptyset \Rightarrow y \notin \mathrm{Dom}(\sigma)$, y entonces:

$$M^\sigma[x \,/\, P^\sigma] = y^\sigma[x \,/\, P^\sigma] = y[x \,/\, P^\sigma] = y = (y[x \,/\, P])^\sigma = (M[x \,/\, P])^\sigma$$

- $M = c \in \mathcal{C}$

  Este caso es trivial pues las sustituciones no afectan a los constructores.

- $M = M_1\, M_2$

$$\begin{array}{rcl} M^\sigma[x \,/\, P^\sigma] & = & (M_1\, M_2)^\sigma[x \,/\, P^\sigma] \\ & \overset{\textsc{Sub-App-Bc}}{=} & (M_1^\sigma\, M_2^\sigma)[x \,/\, P^\sigma] \\ & \overset{\textsc{Sub-App-Bc}}{=} & M_1^\sigma[x \,/\, P^\sigma]\, M_2^\sigma[x \,/\, P^\sigma] \\ & \overset{\text{HI}}{=} & (M_1[x \,/\, P])^\sigma\, (M_2[x \,/\, P])^\sigma \\ & \overset{\textsc{Sub-App-Bc}}{=} & (M_1[x \,/\, P]\, M_2[x \,/\, P])^\sigma \\ & \overset{\textsc{Sub-App-Bc}}{=} & ((M_1\, M_2)[x \,/\, P])^\sigma \\ & = & (M[x \,/\, P])^\sigma \end{array}$$

Notar que la utilización de la hipótesis inductiva queda justificada debido a que $\mathrm{FV}(M_i) \subseteq \mathrm{FV}(M) \Rightarrow \mathrm{Dom}(\sigma) \cap \mathrm{FV}(M_i) = \emptyset$, $i = 1, 2$.

- $M = \lambda y.M_1$

La variable y aparece ligada en M, por lo que, por la convención de Barendregt, asumimos que $y \neq x \wedge y \notin \mathrm{Dom}(\sigma)$. De esta manera,

$$\begin{array}{rcl} M^\sigma[x \,/\, P^\sigma] & = & (\lambda y.M_1)^\sigma[x \,/\, P^\sigma] \\ & \overset{\textsc{Sub-Abs-Bc}}{=} & (\lambda y.M_1^\sigma)\,[x \,/\, P^\sigma] \\ & \overset{\textsc{Sub-Abs-Bc}}{=} & \lambda y.M_1^\sigma[x \,/\, P^\sigma] \\ & \overset{\text{HI}}{=} & \lambda y.(M_1[x \,/\, P])^\sigma \\ & \overset{\textsc{Sub-Abs-Bc}}{=} & (\lambda y.M_1[x \,/\, P])^\sigma \\ & \overset{\textsc{Sub-Abs-Bc}}{=} & ((\lambda y.M_1)[x \,/\, P])^\sigma \\ & = & (M[x \,/\, P])^\sigma \end{array}$$

De manera similar al caso de la aplicación, se puede usar la hipótesis inductiva pues $\mathrm{FV}(M_1) = \mathrm{FV}(M) \Rightarrow \mathrm{Dom}(\sigma) \cap \mathrm{FV}(M_1) = \emptyset$.

- $M = \{\!|\, c_1 \mapsto M_1 \,;\, \cdots \,;\, c_k \mapsto M_k \,|\!\}.N$

$$
\begin{array}{rcl}
M^\sigma[x \,/\, P^\sigma] & = & (\{\!|\, c_1 \mapsto M_1 \,;\, \cdots \,;\, c_k \mapsto M_k \,|\!\}.N)^\sigma[x \,/\, P^\sigma] \\
& \overset{\text{Sub-Case-Bc}}{=} & (\{\!|\, c_1 \mapsto M_1^\sigma \,;\, \cdots \,;\, c_k \mapsto M_k^\sigma \,|\!\}.N^\sigma)\,[x \,/\, P^\sigma] \\
& \overset{\text{Sub-Case-Bc}}{=} & \{\!|\, c_1 \mapsto M_1^\sigma[x \,/\, P^\sigma] \,;\, \cdots \,;\, c_k \mapsto M_k^{\sigma_1}[x \,/\, P^\sigma] \,|\!\}.N^\sigma[x \,/\, P^\sigma] \\
& \overset{\text{HI}}{=} & \{\!|\, c_1 \mapsto (M_1[x \,/\, P])^\sigma \,;\, \cdots \,;\, c_k \mapsto (M_k[x \,/\, P])^\sigma \,|\!\}.(N[x \,/\, P])^\sigma \\
& \overset{\text{Sub-Case-Bc}}{=} & (\{\!|\, c_1 \mapsto M_1[x \,/\, P] \,;\, \cdots \,;\, c_k \mapsto M_k[x \,/\, P] \,|\!\}.N[x \,/\, P])^\sigma \\
& \overset{\text{Sub-Case-Bc}}{=} & ((\{\!|\, c_1 \mapsto M_1 \,;\, \cdots \,;\, c_k \mapsto M_k \,|\!\}.N)[x \,/\, P])^\sigma \\
& = & (M[x \,/\, P])^\sigma
\end{array}
$$

Nuevamente, el uso de la hipótesis inductiva sigue del hecho de que $\mathrm{FV}(M_i) \subseteq \mathrm{FV}(M) \Rightarrow \mathrm{Dom}(\sigma) \cap \mathrm{FV}(M_i) = \emptyset$, $i = 1, \cdots, k+1$, con $M_{k+1} = N$. □

2.8. Comentarios adicionales

El proceso mediante el que se erigió la traducción atravesó varias etapas. En un principio, decidimos adoptar un enfoque ambicioso: el objetivo consistía en buscar soluciones generales que permitiesen traducir términos arbitrarios, tomando como punto de partida las condiciones necesarias para asegurar confluencia de $\lambda\mathcal{C}$ según [45]. A partir del estudio de términos *conflictivos*, en donde las distintas posiciones internas de los patrones en una misma abstracción múltiple[8] corresponden a variables en, al menos, un caso[9], se intentó definir traducciones totales. Las mismas se basaban en diversos mecanismos de preprocesamiento de términos con el objeto de marcar las posiciones de variables de cada patrón con constructores unarios especiales, y luego marcar también esas mismas posiciones en cada instancia de cada patrón. Estas traducciones, además de engorrosas, probaron no ser viables mediante la experimentación, lo cual motivó el replanteo del problema.

A partir de esto, optamos por acotar tan poco como se pueda el dominio de la traducción. Así es como encontramos el conjunto $\mathfrak{M}$, en donde se restringe levemente la condición de no unificabilidad de a pares pero manteniendo el interés y la esencia del cálculo original. Con este nuevo dominio, fue posible definir una traducción (la función Ψ presentada en este capítulo) que sorteó con éxito las pruebas experimentales donde sus predecesores habían fallado.

[8] Asumiendo que dichos patrones comienzan con el mismo constructor.

[9] Tal como sucede, por ejemplo, en $M = \lambda f(x,a,a).x \,|\, \lambda f(b,x,b).x \,|\, \lambda f(c,c,x).x$

Capítulo 3

Un sistema de combinadores para $\lambda\mathcal{B}_\mathcal{C}$

3.1. Introducción

Este capítulo tiene como objetivo formular un sistema de combinadores que capture las características distintivas de $\lambda\mathcal{B}_c$ (i.e., análisis de casos vía case binders), de manera análoga a la lógica combinatoria para el λ-cálculo clásico. Para ello, es fundamental el proceso de eliminación de la abstracción, que en la lógica combinatoria clásica motiva el uso de los combinadores **K** y **S**. En el caso de $\lambda\mathcal{B}_c$, la situación es más compleja debido al hecho de que posee construcciones sintácticas que extienden las del λ-cálculo, para las cuales será también necesario definir nuevos combinadores que permitan "distribuir" los argumentos de las abstracciones en los casos donde dichas construcciones constituyan su cuerpo. Esto, además deviene en la necesidad de definir la aplicación de case binders a términos convencionales.

Estos nuevos combinadores constituyen la base del sistema que presentamos en este capítulo, al que llamaremos $\mathrm{CL}_{\mathcal{B}_c}$. Una particularidad adicional es que posee una cantidad inifinita de reglas de reducción, capturadas por diversos esquemas de reglas. Esta característica es consecuencia de la necesidad de asegurar la confluencia, propiedad esencial sin la cual el cálculo perdería utilidad e interés. Tales esquemas se derivaron a partir de un conjunto inicial de reglas que extendían de manera natural las de la lógica combinatoria clásica, motivados por la necesidad de cerrar los pares críticos allí presentes. Como es sabido, esto es crucial si se quiere lograr la confluencia del sistema.

El capítulo se estructura de la siguiente manera: comentaremos en primera instancia las motivaciones para definir tal sistema en la sección 3.1.1. Luego, mostraremos cómo se definen los términos y case binders de $\mathrm{CL}_{\mathcal{B}_c}$ y los esquemas de reglas asociados en la sección 3.2. La sección 3.3 presentará una serie de definiciones que utilizaremos a menudo, y de ahí abordaremos la prueba de confluencia en la sección 3.4. A continuación veremos cómo representar la abstracción en nuestro sistema (sección 3.5), tal como sucede en CL. Luego daremos traducciones en ambos sentidos para $\mathrm{CL}_{\mathcal{B}_c}$ y $\lambda\mathcal{B}_c$, junto con algunas de sus propiedades (sección 3.6). En la sección 3.7 haremos unos breves comentarios sobre extensionalidad y, por último, discutiremos algunos aspectos adicionales del desarrollo en la sección 3.8.

3.1.1. Motivaciones

Este capítulo introduce un sistema de combinadores asociado al cálculo $\lambda\mathcal{B}_c$. La virtud de un tal sistema proviene de la idea general de algebrización de diversas formas del cálculo lambda, actualmente una tendencia. En este sentido está vigente la idea de que el cálculo lambda es algebraico (ver por ejemplo [64]), es decir, se puede formular adecuadamente en base a cierta estructura algebraica, basada llanamente en operaciones de un álgebra sin necesidad de introducir el concepto de variable ligada. En este caso, se puede utilizar álgebras combinatorias [48, 7], una estructura muy vinculada con las formulaciones categóricas [7]. Las álgebras combinatorias resultan ser una familia de modelos de la lógica combinatoria clásica, a veces incluso extendidas de alguna manera, por ejemplo con objetos adicionales [48].

Cuando se formulan los problemas en lógica combinatoria, éstos son "más algebraicos" y pueden recibir un tratamiento adecuado. Como ejemplo podemos citar el problema de resolubilidad (*solvability*) para el cálculo lambda: dado M un término, determinar si existen $N_1, \cdots, N_n$ tales que $M\,N_1 \cdots N_n =_\beta I = \lambda x.x$. Este problema claramente equivale a: dados M y N términos, determinar si existen $N_1, \cdots, N_n$ tales que $M\,N_1 \cdots N_n =_\beta N$, y representa la posibilidad de "invertir" de algún modo la función que semánticamente está denotada por el término M. Si se formula en lógica combinatoria, cualquier solución allí significa una solución en cálculo lambda. La resolubilidad juega un papel importante en teoría de modelos aplicada al λ-cálculo y guarda relación con la propiedad de separación y los árboles de Böhm. Para una exposición de esta clase de problemas así como de su importancia y aplicaciones, véase por ejemplo [9].

Veamos ahora una de las aplicaciones concretas, y posiblemente la más notable, de contar con una lógica combinatoria para $\lambda\mathcal{B}_c$. Sean M, N términos del cálculo lambda. El problema de unificación en alto orden o semántica (*Higher-Order unification*) es la existencia de una sustitución σ tal que $\sigma M =_{\beta\eta} \sigma N$. Al igual que con primer orden, interesa determinar la existencia de tal unificador σ como así también calcular, de ser posible, el más general.

El problema general es, por supuesto, indecidible. Sin embargo, con ciertas restricciones sobre el conjunto de términos (tipables, por ejemplo) se consigue la posibilidad de tener algoritmos. La unificación (tanto de primer orden como de alto orden) es reconocidamente un problema fundamental en ciencias de la computación, debido a su aplicación directa tanto en el paradigma de programación lógica, en la verificación y compleción de pares críticos y en sistemas de reescritura como en otras áreas muy diversas tales como inteligencia artificial, particularmente aprendizaje automático y reconocimiento de lenguaje natural.

Dicho esto, se observa que resulta complicada la formulación de los algoritmos de unificación de alto orden debido al uso de variables ligadas, que juegan un rol fundamental. Por este motivo, en [25] se propone, dado un problema de unificación como el descripto, la conversión a lógica combinatoria de los términos como paso intermedio, para así suprimir el uso de variables ligadas. De este modo se simplifica considerablemente la formulación del algoritmo, cf. [25].

Incluso el problema puede extenderse cuando se cuenta con una teoría ecuacional de primer orden subyacente [33]. Estos procedimientos en general son no determinísticos, y presentan analogía con el de Martelli-Montanari [49] para primer orden, y el de Huet [28] para alto orden.

Resulta claro entonces que para el cálculo $\lambda\mathcal{B}_c$, para el cual también se pueden plantear problemas de unificación de alto orden, tiene sentido aplicar el mismo procedimiento antedicho. Pero ahora, sobre nuestro sistema de combinadores $\mathrm{CL}_{\mathcal{B}_c}$. Desde ya, este problema y su posible solución están más allá del alcance y objetivos de esta tesis, pero podemos decir

que $\mathrm{CL}_{\mathcal{B}_c}$ muy posiblemente sirva como lenguaje intermedio para buscar soluciones vía un mecanismo similar al anterior,y esto es válido gracias a contar con una conversión ida y vuelta (módulo equivalencia en $\lambda\mathcal{B}_c$), así como gracias al hecho de que $\mathrm{CL}_{\mathcal{B}_c}$ provee una reducción simulada por $\lambda\mathcal{B}_c$ tal como se verá en la sección 3.6. La adaptación del procedimiento general a nuestro escenario representa una posibilidad de trabajo futuro.

Finalmente, otra aplicación de contar con una lógica combinatoria, quizá algo más directa, es la posibilidad de proveer una implementación del cálculo. Ya sea bajo arquitecturas especiales que se basen en dichas operaciones, como bajo una arquitectura convencional, las reglas de la lógica combinatoria no precisan del concepto de variable ligada, con su correspondiente dificultad (control del alcance, renombramiento automático, mantenimiento de asociaciones). Sin ellas, pues, las operaciones son más ligadas a acciones primitivas y en cierto modo más directas. Existe una sólida confirmación de estos hechos en parte debido al trabajo de Turner [66, 67], en el cual se hace una discusión acerca de combinadores clásicos e introduce nuevos que conllevan a una mayor eficiencia.

3.2. Sintaxis y reglas de reducción

Al igual que $\lambda\mathcal{B}_c$, $\mathrm{CL}_{\mathcal{B}_c}$ trabaja con variables y con constructores. Llamaremos $\mathcal{V}$ al conjunto inifinito numerable de variables (notadas x, y, z, etc.), y $\mathcal{C}$ al de constructores (notados c, d, etc.), asumiendo que ambos son disjuntos. Los términos (M, N, etc.) y case binders (θ, ϕ, etc.) de este sistema se definen en forma inductiva tal como se muestra a continuación.

Términos	M, N	$::=$	x	$(x \in \mathcal{V})$
		$\mid$	c	$(c \in \mathcal{C})$
		$\mid$	$\mathbf{K}$	
		$\mid$	$\mathbf{S}$	
		$\mid$	MN	
		$\mid$	$\theta.M$	
		$\mid$	$\mathbf{A}^n\,\theta$	$(n \in \mathbb{N})$
Case binders	θ	$::=$	$\{\!\mid c_1 \mapsto M_1; \cdots ; c_n \mapsto M_n \mid\!\}$	$(c_i \neq c_j$ si $i \neq j)$
		$\mid$	$\theta \bullet M$	
		$\mid$	$\mathbf{B}^n\,\theta$	$(n \in \mathbb{N})$

Para cada natural n, cada símbolo $\mathbf{A}^n$ y cada símbolo $\mathbf{B}^n$ representan los nuevos combinadores ya comentados en la sección 3.1. Dichos combinadores fueron motivados por la necesidad de representar la abstracción en este sistema, tal como sucede en la lógica combinatoria clásica. Notar, además, que $\mathrm{CL}_{\mathcal{B}_c}$ introduce nuevos case binders a los clásicos de $\lambda\mathcal{B}_c$, lo cual se vio motivado también por los motivos ya mencionados. Intuitivamente, el símbolo $\bullet$ representa la aplicación de un case binder a un término; tal construcción es necesaria para distribuir el argumento de una función dentro de un case binder. Abordaremos nuevamente estos temas en la sección 3.5.1, en donde explicaremos en mayor detalle el propósito de estos combinadores y de las nuevas construcciones.

$\mathrm{CL}_{\mathcal{B}_c}$ posee una cantidad infinita de reglas de reducción que, no obstante, pueden representarse a través de *esquemas* de reglas. El sistema, entonces, se define a través de nueve reglas y esquemas de reglas de reducción, mostrados en la Figura 3.1. Estos esquemas de

reglas, a diferencia de las reglas usuales, engloban una *familia* de reglas de reducción. Por ejemplo, cada esquema que involucre vectores de case binders (ver Definición 3.3.3) define en realidad una regla distinta para cada vector. Lo mismo sucede con los combinadores $\mathbf{A}^n$ y $\mathbf{B}^n$: para cada natural n, existirá una regla asociada.

Llamaremos $\to_{\mathrm{CL}_{\mathcal{B}_c}}$ a la unión de todas las reglas de reducción del sistema. Si el contexto no es ambiguo, también llamaremos simplemente $\to$ a tal reducción.

(K)	$\vec{\theta}.(\vec{\phi}.\mathbf{K}\,M)\,N$	$\to$	$\vec{\theta}.\vec{\phi}.M$	
(S)	$\vec{\theta}.(\vec{\phi}.(\vec{\omega}.\mathbf{S}\,M)\,N)\,P$	$\to$	$\vec{\theta}.\vec{\phi}.\vec{\omega}.(M\,P\,(N\,P))$	
(CaseCons)	$\theta.c$	$\to$	M	$(c \mapsto M \in \theta)$
(CaseApp)	$\theta.(M\,N)$	$\to$	$\theta.M\,N$	
(DotCase)	$\{\!\|\, c_i \mapsto M_i \,\|\!\}_{i=1}^n \bullet N$	$\to$	$\{\!\|\, c_i \mapsto M_i N \,\|\!\}_{i=1}^n$	
(A-Dist)	$\vec{\theta}.(\vec{\phi}.\mathbf{A}^0\,\varphi\,M)\,N$	$\to$	$\vec{\theta}.\vec{\phi}.((\varphi \bullet N).(M\,N))$	
(A-Dec)	$\vec{\theta}.\mathbf{A}^{n+1}\,\varphi\,M$	$\to$	$\vec{\theta}.(\mathbf{A}^n\,(\varphi \bullet M))$	
(B-Dist)	$\mathbf{B}^0\,\varphi \bullet M \bullet N$	$\to$	$(\varphi \bullet N) \bullet (M\,N)$	
(B-Dec)	$\mathbf{B}^{n+1}\,\varphi \bullet M$	$\to$	$\mathbf{B}^n\,(\varphi \bullet M)$	

Figura 3.1: Reglas (y esquemas de reglas) de reducción de $\mathrm{CL}_{\mathcal{B}_c}$

A modo de ejemplo, consideremos el término $\mathbf{K}\,\mathbf{S}\,f$. El esquema K permite ser instanciado (tomando $\vec{\theta} = \vec{\phi} = \emptyset$) de manera tal que

$$\mathbf{K}\,\mathbf{S}\,f \to_{\mathrm{CL}_{\mathcal{B}_c}} \mathbf{S}$$

De igual manera, si uno tuviera el término $\theta_1.(\theta_2.\theta_3.\mathbf{K}\,\mathbf{S})\,f$, el mismo esquema permitiría concluir que

$$\theta_1.(\theta_2.\theta_3.\mathbf{K}\,\mathbf{S})\,f \to_{\mathrm{CL}_{\mathcal{B}_c}} \theta_1.\theta_2.\theta_3.\mathbf{S}$$

Respecto de la regla CaseCons, notemos que sólo puede ser utilizada si θ es un case binder clásico (i.e., un conjunto).

3.3. Definiciones previas

En esta sección presentaremos una serie de definiciones que serán de utilidad durante el desarrollo del capítulo.

Definición 3.3.1.

- El *conjunto de términos* de $\mathrm{CL}_{\mathcal{B}_c}$ lo notaremos $\mathbf{CL}^{\mathrm{T}}_{\mathcal{B}_\mathbf{c}}$.
- El *conjunto de case binders* de $\mathrm{CL}_{\mathcal{B}_c}$ lo notaremos $\mathbf{CL}^{\mathrm{CB}}_{\mathcal{B}_\mathbf{c}}$.
- La unión de ambos conjuntos la notaremos $\mathbf{CL}_{\mathcal{B}_\mathbf{c}}$.

Definición 3.3.2. El *conjunto de átomos* de $\mathrm{CL}_{\mathcal{B}_c}$es el conjunto

$$\mathcal{A} = \mathcal{V} \cup \mathcal{C} \cup \{\mathbf{K}, \mathbf{S}\}$$

Definición 3.3.3.

- Un *vector de case binders* es una secuencia de k case binders $\vec{\theta} = \theta_1 \cdots \theta_k$, con $k \in \mathbb{N}$. El conjunto de tales vectores viene dado por la gramática

$$\vec{\theta} \quad ::= \quad \emptyset \quad | \quad \vec{\theta}\,\theta$$

  Utilizaremos las letras $\theta, \phi, \varphi, \omega$ y τ para referinos a case binders y a vectores de case binders.

- La *longitud* de $\vec{\theta}$ se nota $|\vec{\theta}|$ y se define así:

$$|\vec{\theta}| = \begin{cases} 0 & \text{si } \vec{\theta} = \emptyset \\ 1 + |\vec{\phi}| & \text{si } \vec{\theta} = \vec{\phi}\varphi \end{cases}$$

- Sea $M \in \mathbf{CL}^{\mathrm{T}}_{\mathcal{B}_\mathbf{c}}$ y sea $\vec{\theta} = \theta_1 \cdots \theta_k$ un vector de case binders. La expresión $\vec{\theta}.M$ es azúcar sintáctico para $\theta_1.(\theta_2.(\cdots(\theta_k.M)\cdots)) = \theta_1.\theta_2.\cdots.\theta_k.M$ [1].

- Dada una relación de reducción simultánea $\rightarrow$ y dos vectores de case binders $\vec{\theta} = \theta_1 \cdots \theta_k$ y $\vec{\phi} = \phi_1 \cdots \phi_k$, notaremos con $\vec{\theta} \rightarrow \vec{\phi}$ la reducción simultánea $\theta_i \rightarrow \phi_i$, para cada $i = 1, \cdots, k$.

Definición 3.3.4. Sea $M \in \mathbf{CL}_{\mathcal{B}_\mathbf{c}}$. El conjunto de *variables* de M se nota $\mathrm{FV}(M)$ y se define de manera inductiva como muestra la Figura 3.2.

(FV-Atom)	$\mathrm{FV}(a)$	$=$	$\begin{cases} \{a\} & \text{si } a \in \mathcal{V} \\ \emptyset & \text{en otro caso} \end{cases}$ $(a \in \mathcal{A})$
(FV-App)	$\mathrm{FV}(M_1\,M_2)$	$=$	$\mathrm{FV}(M_1) \cup \mathrm{FV}(M_2)$
(FV-A)	$\mathrm{FV}(\mathbf{A}^n\,\theta)$	$=$	$\mathrm{FV}(\theta)$
(FV-CaseTerm)	$\mathrm{FV}(\theta.M_1)$	$=$	$\mathrm{FV}(\theta) \cup \mathrm{FV}(M_1)$
(FV-Case)	$\mathrm{FV}(\{\!\|\, c_i \mapsto M_i \,\|\!\}_{i=1}^n)$	$=$	$\bigcup_{i=1}^{n} \mathrm{FV}(M_i)$
(FV-Dot)	$\mathrm{FV}(\theta \bullet M_1)$	$=$	$\mathrm{FV}(\theta) \cup \mathrm{FV}(M_1)$
(FV-B)	$\mathrm{FV}(\mathbf{B}^n\,\theta)$	$=$	$\mathrm{FV}(\theta)$

Figura 3.2: Variables libres para términos de $\mathrm{CL}_{\mathcal{B}_c}$

Definición 3.3.5. Sea $M, N \in \mathbf{CL}_{\mathcal{B}_\mathbf{c}}$ y $x \in \mathcal{V}$. Definimos la *sustitución* en M de la variable x por el término N por inducción en M, como muestra la figura 3.3.

Definición 3.3.6. Sea $\theta = \{\!|\, c_i \mapsto M_i \,|\!\}_{i=1}^n$ un case binder clásico de $\lambda\mathcal{B}_c$.

- Sea $M \in \Lambda\mathcal{B}_c$. Definimos un meta-operador $\circ$ de *aplicación simultánea*:

$$\{\!|\, c_i \mapsto M_i \,|\!\}_{i=1}^n \circ M = \{\!|\, c_i \mapsto M_i\, M \,|\!\}_{i=1}^n$$

[1] Observar que $\vec{\theta}.M = M$ sólo si $\vec{\theta} = \emptyset$

(Sub-Atom)	$a[x \,/\, N]$	$=$	$\begin{cases} N & \text{si } a = x \\ a & \text{en otro caso} \end{cases}$	$(a \in \mathcal{A})$
(Sub-App)	$(M_1\, M_2)[x \,/\, N]$	$=$	$M_1[x \,/\, N]\, M_2[x \,/\, N]$	
(Sub-A)	$(\mathbf{A}^n\, \theta)[x \,/\, N]$	$=$	$\mathbf{A}^n\, \theta[x \,/\, N]$	
(Sub-CaseTerm)	$(\theta.M_1)[x \,/\, N]$	$=$	$\theta[x \,/\, N].M_1[x \,/\, N]$	
(Sub-Case)	$(\{\!\| c_i \mapsto M_i \|\!\}_{i=1}^n)[x \,/\, N]$	$=$	$\{\!\| c_i \mapsto M_i[x \,/\, N] \|\!\}_{i=1}^n$	
(Sub-Dot)	$(\theta \bullet M_1)[x \,/\, N]$	$=$	$\theta[x \,/\, N] \bullet M_1[x \,/\, N]$	
(Sub-B)	$(\mathbf{B}^n\, \theta)[x \,/\, N]$	$=$	$\mathbf{B}^n\, \theta[x \,/\, N]$	

Figura 3.3: Sustitución de variables en $\mathrm{CL}_{\mathcal{B}_c}$

- Sea $x \in \mathcal{V}$. El meta-operador $\underline{\lambda}$ de *abstracción simultánea* se define así:

$$\underline{\lambda}x.\{\!\| c_i \mapsto M_i \|\!\}_{i=1}^n = \{\!\| c_i \mapsto \lambda x.M_i \|\!\}_{i=1}^n$$

  Generalizado a n variables,

$$\underline{\lambda}x_1 \cdots x_n.\theta = \underline{\lambda}x_1.(\underline{\lambda}x_2.\cdots(\underline{\lambda}x_n.\theta)\cdots)$$

3.4. Confluencia de $\mathrm{CL}_{\mathcal{B}_c}$

En primer lugar, mostraremos que el sistema es confluente. La confluencia, desde ya, es una propiedad fundamental dado que garantiza la unicidad de cómputos, independientemente del orden de evaluación de los términos. La metodología desarrollada en esta sección corresponde a Tait-Martin-Löf [9]: en esencia, la prueba se descompone en cuatro etapas resumidas a continuación.

1. Definir una *reducción paralela* $\rightrightarrows$ a partir de la relación de reducción original $\rightarrow_{\mathrm{CL}_{\mathcal{B}_c}}$ (sección 3.4.1). Como su nombre lo indica, tal reducción opera en forma paralela, esto es, permite contraer simultáneamente redexes ubicados en posiciones paralelas en los términos.

2. Demostrar que $\rightrightarrows$ satisface $\diamond$ (Proposición 3.4.1).

3. Demostrar que cada paso de reducción en $\mathrm{CL}_{\mathcal{B}_c}$ puede simularse en un paso de $\rightrightarrows$ y que, a su vez, cada paso de $\rightrightarrows$ puede simularse en cero o más pasos de $\rightarrow_{\mathrm{CL}_{\mathcal{B}_c}}$ (Proposición 3.4.2).

4. Concluir que el sistema es confluente (Proposición 3.4.3).

En la siguiente sección se presenta la reducción paralela $\rightrightarrows$. Luego, en la subsección 3.4.1.1, se demuestran las Proposiciones anteriormente mencionadas.

3.4.1. La reducción paralela $\rightrightarrows$

La Figura 3.4 detalla las reglas que definen la reducción $\rightrightarrows$. Obsérvese que ésta se construye a partir de $\rightarrow_{\mathrm{CL}_{\mathcal{B}_c}}$, agregando reglas adicionales (y extendiendo algunas existentes) de manera de permitir contracciones de reductos en posiciones disjuntas.

$$\frac{}{M \rightrightarrows M}\ (\rightrightarrows\text{R}) \qquad \frac{\vec{\theta} \rightrightarrows \vec{\theta'}\ \ \vec{\phi} \rightrightarrows \vec{\phi'}}{\vec{\theta}.(\vec{\phi}.\mathbf{K}\,M)\,N \rightrightarrows \vec{\theta'}.\vec{\phi'}.M}\ (\rightrightarrows\text{K})$$

$$\frac{\vec{\theta} \rightrightarrows \vec{\theta'}\ \ \vec{\phi} \rightrightarrows \vec{\phi'}\ \ \vec{\omega} \rightrightarrows \vec{\omega'}}{\vec{\theta}.(\vec{\phi}.(\vec{\omega}.\mathbf{S}\,M)\,N)\,P \rightrightarrows \vec{\theta'}.\vec{\phi'}.\vec{\omega'}.(M\,P\,(N\,P))}\ (\rightrightarrows\text{S})$$

$$\frac{c \mapsto M \in \theta}{\theta.c \rightrightarrows M}\ (\rightrightarrows\textsc{CaseCons}) \qquad \frac{}{\theta.(M\,N) \rightrightarrows \theta.M\,N}\ (\rightrightarrows\textsc{CaseApp})$$

$$\frac{}{\{\!|\, c_i \mapsto M_i \,|\!\}_{i=1}^n \bullet N \rightrightarrows \{\!|\, c_i \mapsto M_i N \,|\!\}_{i=1}^n}\ (\rightrightarrows\textsc{DotCase})$$

$$\frac{\vec{\theta} \rightrightarrows \vec{\theta'}\ \ \vec{\phi} \rightrightarrows \vec{\phi'}}{\vec{\theta}.(\vec{\phi}.\mathbf{A}^0\,\varphi\,M)\,N \rightrightarrows \vec{\theta'}.\vec{\phi'}.((\varphi \bullet N).(M\,N))}\ (\rightrightarrows\text{A-}\textsc{Dist})$$

$$\frac{}{\mathbf{B}^0\,\varphi \bullet M \bullet N \rightrightarrows (\varphi \bullet N) \bullet (M\,N)}\ (\rightrightarrows\text{B-}\textsc{Dist})$$

$$\frac{\vec{\theta} \rightrightarrows \vec{\theta'}}{\vec{\theta}.\mathbf{A}^{n+1}\,\varphi\,M \rightrightarrows \vec{\theta'}.\mathbf{A}^n\,(\varphi \bullet M)}\ (\rightrightarrows\text{A-}\textsc{Dec})$$

$$\frac{}{\mathbf{B}^{n+1}\,\varphi \bullet M \rightrightarrows \mathbf{B}^n\,(\varphi \bullet M)}\ (\rightrightarrows\text{B-}\textsc{Dec})$$

$$\frac{\theta \rightrightarrows \theta'}{\mathbf{A}^n\theta \rightrightarrows \mathbf{A}^n\theta'}\ (\rightrightarrows\textsc{ACase}) \qquad \frac{\theta \rightrightarrows \theta'}{\mathbf{B}^n\theta \rightrightarrows \mathbf{B}^n\theta'}\ (\rightrightarrows\textsc{BCase})$$

$$\frac{M_i \rightrightarrows M_i' \quad (1 \le i \le n)}{\{\!|\, c_i \mapsto M_i \,|\!\}_{i=1}^n \rightrightarrows \{\!|\, c_i \mapsto M_i' \,|\!\}_{i=1}^n}\ (\rightrightarrows\textsc{Case}) \qquad \frac{M \rightrightarrows M'\ \ N \rightrightarrows N'}{M\,N \rightrightarrows M'\,N'}\ (\rightrightarrows\textsc{App})$$

$$\frac{\theta \rightrightarrows \theta'\ \ M \rightrightarrows M'}{\theta.M \rightrightarrows \theta'.M'}\ (\rightrightarrows\textsc{CaseTerm}) \qquad \frac{\theta \rightrightarrows \theta'\ \ M \rightrightarrows M'}{\theta \bullet M \rightrightarrows \theta' \bullet M'}\ (\rightrightarrows\textsc{CaseDot})$$

Figura 3.4: La reducción paralela $\rightrightarrows$

3.4.1.1. Propiedades de $\rightrightarrows$

Pasemos ahora a probar las proposiciones que nos permitirán asegurar la confluencia de $\mathrm{CL}_{\mathcal{B}_c}$. Primeramente, mostraremos que $\rightrightarrows$ satisface $\diamond$.

Proposición 3.4.1. $\rightrightarrows$ satisface $\diamond$

Demostración Sean $M, M_1, M_2 \in \mathbf{CL}_{\mathcal{B}_\mathbf{c}}$ tales que $M_2 \leftleftarrows M \rightrightarrows M_1$. Para probar que $\rightrightarrows$ satisface $\diamond$, mostraremos por inducción en $M \rightrightarrows M_1$ que $\exists\, M_3 \in \mathbf{CL}_{\mathcal{B}_\mathbf{c}} \,/\, M_1 \rightrightarrows M_3 \leftleftarrows M_2$. En cada caso, analizaremos las diferentes formas en las que $M \rightrightarrows M_2$ y, para cada una de ellas, encontraremos un M_3 que satisfaga lo antedicho. Los casos donde $M \rightrightarrows M_2$ a través de $\rightrightarrows$R no se considerarán ya que cierran trivialmente.

- $M \rightrightarrows M = M_1$ ($\rightrightarrows$R)

  Es suficiente tomar $M_3 = M_2$.

- $M = \vec{\theta}.(\vec{\phi}.\mathbf{K}\, N_1)\, N_2 \rightrightarrows \vec{\theta}'.\vec{\phi}'.N_1 = M_1$ (si $\vec{\theta} \rightrightarrows \vec{\theta}'$ y $\vec{\phi} \rightrightarrows \vec{\phi}'$) ($\rightrightarrows$K)

  En primer lugar, supongamos que $|\vec{\theta}| > 0$. De ser así, $\vec{\theta} = \vec{\omega}\,\varphi$ y, luego,

$$\begin{array}{rcl} M & = & \vec{\theta}.(\vec{\phi}.\mathbf{K}\, N_1)\, N_2 \\ & = & \vec{\omega}.\varphi.(\vec{\phi}.\mathbf{K}\, N_1)\, N_2 \\ & \overset{\rightrightarrows\textsc{CaseApp}}{\rightrightarrows} & M_2 \\ & = & \vec{\omega}.(\varphi.\vec{\phi}.\mathbf{K}\, N_1)\, N_2 \\ & \overset{\rightrightarrows\mathrm{K}}{\rightrightarrows} & \vec{\omega}'.\varphi'.\vec{\phi}'.N_1 \\ & = & M_1 \end{array}$$

  en donde $\varphi \rightrightarrows \varphi'$, $\vec{\omega} \rightrightarrows \vec{\omega}'$ y $\vec{\theta}' = \vec{\omega}'\,\varphi'$. De este desarrollo, se obtiene que basta tomar $M_3 = M_1$.

  Por otro lado, la única alternativa restante a tener en cuenta es que $M \rightrightarrows M_2$ mediante $\rightrightarrows$App. Supongamos entonces que $N_2 \rightrightarrows N_2''$ y que $\vec{\theta}.(\vec{\phi}.\mathbf{K}\, N_1) \rightrightarrows P$. Obviando el caso donde P se obtiene a través de $\rightrightarrows$CaseApp (sería muy similar al anterior), P sólo podrá tener la forma $\vec{\theta}''.(\vec{\phi}''.\mathbf{K}\, N_1'')$, con $\vec{\theta} \rightrightarrows \vec{\theta}''$, $\vec{\phi} \rightrightarrows \vec{\phi}''$ y $N_1 \rightrightarrows N_1''$. Por hipótesis inductiva, deben existir $\vec{\theta}'''$ y $\vec{\phi}'''$ tales que $\vec{\theta}' \rightrightarrows \vec{\theta}''' \leftleftarrows \vec{\theta}''$ y $\vec{\phi}' \rightrightarrows \vec{\phi}''' \leftleftarrows \vec{\phi}''$ [2]. Luego,

$$\begin{array}{rcl} M & = & \vec{\theta}.(\vec{\phi}.\mathbf{K}\, N_1)\, N_2 \\ & \overset{\rightrightarrows\textsc{App}}{\rightrightarrows} & M_2 \\ & = & \vec{\theta}''.(\vec{\phi}''.\mathbf{K}\, N_1'')\, N_2'' \\ & \overset{\rightrightarrows\mathrm{K}}{\rightrightarrows} & \vec{\theta}'''.\vec{\phi}'''.N_1'' \\ & \overset{\rightrightarrows\textsc{CaseTerm}}{\leftleftarrows} & M_1 \end{array}$$

  De esta manera, $M_3 = \vec{\theta}'''.\vec{\phi}'''.N_1''$.

- $M = \vec{\theta}.(\vec{\phi}.(\vec{\omega}.\mathbf{S}\, N_1)\, N_2)\, N_3 \rightrightarrows \vec{\theta}'.\vec{\phi}'.\vec{\omega}'.(N_1\, N_3\, (N_2\, N_3)) = M_1$ (si $\vec{\theta} \rightrightarrows \vec{\theta}'$, $\vec{\phi} \rightrightarrows \vec{\phi}'$ y $\vec{\omega} \rightrightarrows \vec{\omega}'$) ($\rightrightarrows$S)

[2] De ser $|\vec{\theta}| = |\vec{\phi}| = 0$, estas hipótesis inductivas no tendrían lugar. El razonamiento que sigue, no obstante, es análogo a lo que ocurriría en tal caso.

Razonando en forma similar al caso previo, supongamos primero que $|\vec{\theta}| > 0$, lo cual implica que $\vec{\theta} = \vec{\tau}\,\varphi$. Entonces,

$$\begin{array}{rcl} M & = & \vec{\theta}.(\vec{\phi}.(\vec{\omega}.\mathbf{S}\,N_1)\,N_2)\,N_3 \\ & = & \vec{\tau}.\varphi.(\vec{\phi}.(\vec{\omega}.\mathbf{S}\,N_1)\,N_2)\,N_3 \\ & \overset{\Rightarrow\textsc{CaseApp}}{\rightrightarrows} & M_2 \\ & = & \vec{\tau}.(\varphi.\vec{\phi}.(\vec{\omega}.\mathbf{S}\,N_1)\,N_2)\,N_3 \\ & \overset{\Rightarrow\mathrm{S}}{\rightrightarrows} & \vec{\tau}'.\varphi'.\vec{\phi}'.\vec{\omega}'.(N_1\,N_3\,(N_2\,N_3)) \\ & = & M_1 \end{array}$$

donde $\varphi \rightrightarrows \varphi'$, $\vec{\tau} \rightrightarrows \vec{\tau}'$ y $\vec{\theta}' = \vec{\tau}'\,\varphi'$. Así, $M_3 = M_1$.

Ahora bien, supongamos que $|\vec{\phi}| > 0$. Luego, $\vec{\phi} = \vec{\tau}\,\varphi$ y entonces,

$$\begin{array}{rcl} M & = & \vec{\theta}.(\vec{\phi}.(\vec{\omega}.\mathbf{S}\,N_1)\,N_2)\,N_3 \\ & = & \vec{\theta}.(\vec{\tau}.\varphi.(\vec{\omega}.\mathbf{S}\,N_1)\,N_2)\,N_3 \\ & \overset{\Rightarrow\textsc{CaseApp}}{\rightrightarrows} & M_2 \\ & = & \vec{\theta}.(\vec{\tau}.(\varphi.\vec{\omega}.\mathbf{S}\,N_1)\,N_2)\,N_3 \\ & \overset{\Rightarrow\mathrm{S}}{\rightrightarrows} & \vec{\theta}.\vec{\tau}'.\varphi'.\vec{\omega}'.(N_1\,N_3\,(N_2\,N_3)) \\ & = & M_1 \end{array}$$

i.e., $M_3 = M_1$ una vez más.

La última posibilidad es que M_2 se obtenga al aplicar la regla $\Rightarrow$App. De manera análoga a lo argumentado para el caso de $\Rightarrow$K, tenemos que $M_2 = \vec{\theta}''.(\vec{\phi}''.(\vec{\omega}''.\mathbf{S}\,N_1'')\,N_2'')\,N_3''$ donde $N_3 \rightrightarrows N_3''$, $N_2 \rightrightarrows N_2''$, $N_1 \rightrightarrows N_1''$, $\vec{\theta} \rightrightarrows \vec{\theta}''$, $\vec{\phi} \rightrightarrows \vec{\phi}''$ y $\vec{\omega} \rightrightarrows \vec{\omega}''$. Por hipótesis inductiva, existen $\vec{\theta}''', \vec{\phi}''', \vec{\omega}'''$ tales que $\vec{\theta}' \rightrightarrows \vec{\theta}''' \leftleftarrows \vec{\theta}''$, $\vec{\phi}' \rightrightarrows \vec{\phi}''' \leftleftarrows \vec{\phi}''$ y $\vec{\omega}' \rightrightarrows \vec{\omega}''' \leftleftarrows \vec{\omega}''$.En consecuencia,

$$\begin{array}{rcl} M & = & \vec{\theta}.(\vec{\phi}.(\vec{\omega}.\mathbf{S}\,N_1)\,N_2)\,N_3 \\ & \overset{\Rightarrow\textsc{App}}{\rightrightarrows} & M_2 \\ & = & \vec{\theta}''.(\vec{\phi}''.(\vec{\omega}''.\mathbf{S}\,N_1'')\,N_2'')\,N_3'' \\ & \overset{\Rightarrow\mathrm{S}}{\rightrightarrows} & \vec{\theta}'''.\vec{\phi}'''.\vec{\omega}'''.(N_1''\,N_3''\,(N_2''\,N_3'')) \\ & \leftleftarrows & M_1 \end{array}$$

La última reducción se deriva por medio de las hipótesis y de sucesivas aplicaciones de las reglas $\Rightarrow$App y $\Rightarrow$CaseTerm. Se concluye que basta tomar $M_3 = \vec{\theta}'''.\vec{\phi}'''.\vec{\omega}'''.(N_1''\,N_3''\,(N_2''\,N_3''))$. Los casos en donde $\vec{\theta}.(\vec{\phi}.(\vec{\omega}.\mathbf{S}\,N_1)\,N_2)$ reduce mediante $\Rightarrow$CaseApp se resuelven de manera similar a lo primero, razón por la cual son omitidos [3].

- $M = \theta.c \rightrightarrows M_1$ (si $c \mapsto M_1 \in \theta$) ($\Rightarrow$CaseCons)

En tal situación, debe ocurrir que $\theta = \{\!|\, c_i \mapsto N_i \,|\!\}_{i=1}^{n}$ para algún $n \in \mathbb{N}$ y, además, $c = c_j$ y $M_1 = N_j$ para cierto $j = 1, \cdots, n$. La reducción $M \rightrightarrows M_2$ sólo puede darse a través de $\Rightarrow$CaseTerm, y es tal que $M_2 = \theta'.c$, con $\theta \rightrightarrows \theta'$. Luego, deben existir términos

[3] La diferencia yace en que será preciso reducir las dos apariciones de N_3 en M_1 a N_3'', de manera que M_1 debe reducir en un paso a otro término donde se refleje esto.

$N'_1, \cdots, N'_n$ tales que $N_i \rightrightarrows N'_i$ y $\theta' = \{\!|\, c_i \mapsto N'_i \,|\!\}_{i=1}^{n}$. En consecuencia,

$$\begin{array}{rcl} M & = & \theta.c \\ & \underset{\rightrightarrows \textsc{CaseTerm}}{} & \\ & \rightrightarrows & M_2 \\ & = & \theta'.c \\ & = & \{\!|\, c_1 \mapsto N'_1; \cdots; c_n \mapsto N'_n \,|\!\}.c_j \\ & \underset{\rightrightarrows \textsc{CaseCons}}{} & \\ & \rightrightarrows & N'_j \\ & \leftleftarrows & N_j \\ & = & M_1 \end{array}$$

Es suficiente, entonces, tomar $M_3 = N'_j$.

- $M = \theta.(N_1\, N_2) \rightrightarrows \theta.N_1\, N_2 = M_1$ ($\rightrightarrows$CaseApp)

Este escenario requiere estudiar varios casos posibles. En lo que sigue, cada aparición de $\vec{\alpha}'$ es tal que $\vec{\alpha} \rightrightarrows \vec{\alpha}'$, donde $\alpha = \varphi, \phi, \omega$.

(i) $M = \theta.(\vec{\phi}.(\vec{\omega}.\mathbf{K}\, P)\, N_2)$

$$\begin{array}{rcl} M & = & \theta.(\vec{\phi}.(\vec{\omega}.\mathbf{K}\, P)\, N_2) \\ & \underset{\rightrightarrows \mathrm{K}}{} & \\ & \rightrightarrows & M_2 \\ & = & \theta.\vec{\phi}'.\vec{\omega}'.P \\ & \underset{\rightrightarrows \mathrm{K}}{} & \\ & \leftleftarrows & \theta.\vec{\phi}.(\vec{\omega}.\mathbf{K}\, P)\, N_2 \\ & = & M_1 \end{array}$$

$\Rightarrow M_3 = M_2$. Observar que, por $\rightrightarrows$R, $\theta \rightrightarrows \theta$.

(ii) $M = \theta.(\vec{\phi}.(\vec{\omega}.(\vec{\varphi}.\mathbf{S}\, P)\, Q)\, N_2)$

$$\begin{array}{rcl} M & = & \theta.(\vec{\phi}.(\vec{\omega}.(\vec{\varphi}.\mathbf{S}\, P)\, Q)\, N_2) \\ & \underset{\rightrightarrows \mathrm{S}}{} & \\ & \rightrightarrows & M_2 \\ & = & \theta.\vec{\phi}'.\vec{\omega}'.\vec{\varphi}'.(P\, N_2\, (Q\, N_2)) \\ & \underset{\rightrightarrows \mathrm{S}}{} & \\ & \leftleftarrows & \theta.\vec{\phi}.(\vec{\omega}.(\vec{\varphi}.\mathbf{S}\, P)\, Q)\, N_2 \\ & = & M_1 \end{array}$$

$\Rightarrow M_3 = M_2$.

(iii) $M = \theta.(\vec{\phi}.(\vec{\omega}.\mathbf{A}^0\, \varphi\, P)\, N_2)$
El mecanismo es análogo al del ítem (i).

(iv) $M = \theta.(\vec{\phi}.\mathbf{A}^{n+1}\, \varphi\, N_2)$

$$\begin{array}{rcl} M & = & \theta.(\vec{\phi}.\mathbf{A}^{n+1}\, \varphi\, N_2) \\ & \underset{\rightrightarrows \text{A-Dec}}{} & \\ & \rightrightarrows & M_2 \\ & = & \theta.\vec{\phi}'.\mathbf{A}^{n}\, (\varphi \bullet N_2) \\ & \underset{\rightrightarrows \text{A-Dec}}{} & \\ & \leftleftarrows & \theta.\vec{\phi}.\mathbf{A}^{n+1}\, \varphi\, N_2 \\ & = & M_1 \end{array}$$

$\Rightarrow M_3 = M_2$.

(v) Reducción a M_2 utilizando la regla $\rightrightarrows$CaseTerm.
El caso donde $N_1\, N_2 \rightrightarrows N_1'\, N_2'$ es inmediato. Supongamos, entonces, que $N_1\, N_2 \rightrightarrows P \neq N_1'\, N_2'$. En esta situación, $M_2 = \phi.P$, siendo ϕ tal que $\theta \rightrightarrows \phi$. Los subcasos a tener en cuenta son exactamente los mismos de los ítems anteriores, con la diferencia de que, ahora, el case binder aparece reducido. El mecanismo de la prueba es esencialmente el mismo; en lugar de utilizar $\rightrightarrows$R para derivar $\theta \rightrightarrows \theta$ en la reducción de M_1 a M_2, se usa directamente que $\theta \rightrightarrows \phi$.

- $M = \{\!|\, c_i \mapsto N_i \,|\!\}_{i=1}^n \bullet N \rightrightarrows \{\!|\, c_i \mapsto N_i N \,|\!\}_{i=1}^n = M_1$ ($\rightrightarrows$DotCase)

En esta oportunidad, sólo debemos considerar el caso en el que M_2 se obtiene a través de $\rightrightarrows$CaseDot. Supongamos que $N \rightrightarrows N'$ y, para cada $i = 1, \cdots, n$, $N_i \rightrightarrows N_i'$. Entonces,

$$\begin{array}{lcl} M & = & \{\!|\, c_i \mapsto N_i \,|\!\}_{i=1}^n \bullet N \\ & \underset{\rightrightarrows \text{CaseDot}}{\rightrightarrows} & M_2 \\ & = & \{\!|\, c_i \mapsto N_i' \,|\!\}_{i=1}^n \bullet N' \\ & \underset{\rightrightarrows \text{DotCase}}{\rightrightarrows} & \{\!|\, c_i \mapsto N_i' N' \,|\!\}_{i=1}^n \\ & \leftrightarrows & \{\!|\, c_i \mapsto N_i N \,|\!\}_{i=1}^n \\ & = & M_1 \end{array}$$

La última reducción se deriva aplicando reiteradas veces la regla $\rightrightarrows$App y luego la regla $\rightrightarrows$Case. De esto, sigue que $M_3 = \{\!|\, c_i \mapsto N_i' N' \,|\!\}_{i=1}^n$.

- $M = \vec{\theta}.(\vec{\phi}.\mathbf{A}^0\, \varphi\, N_1)\, N_2 \rightrightarrows \vec{\theta}'.\vec{\phi}'.((\varphi \bullet N_2).(N_1\, N_2)) = M_1$ (si $\vec{\theta} \rightrightarrows \vec{\theta}'$ y $\vec{\phi} \rightrightarrows \vec{\phi}'$) ($\rightrightarrows$A-Dist)

Este caso es análogo al de $\rightrightarrows$K. Existe, no obstante, un caso extra a tener en cuenta, y es cuando M_2 se deriva aplicando la regla $\rightrightarrows$ACase y luego, en forma sucesiva, las reglas $\rightrightarrows$R, $\rightrightarrows$App y $\rightrightarrows$CaseTerm. Sin embargo, se trata de un caso sencillo y, en consecuencia, será omitido.

- $M = \mathbf{B}^0\, \varphi \bullet N_1 \bullet N_2 \rightrightarrows (\varphi \bullet N_2) \bullet (N_1\, N_2) = M_1$ ($\rightrightarrows$B-Dist)

En esta oportunidad, puede suceder que $M \rightrightarrows M_2$ por $\rightrightarrows$CaseDot o por $\rightrightarrows$BCase [4]. El primer caso conforma el escenario más general posible (el otro puede verse como instancia de él). Por ende, supongamos que $N_2 \rightrightarrows N_2'$, y que $\mathbf{B}^0\, \varphi \bullet N_1 \rightrightarrows P$. Necesariamente debe ocurrir que $P = \mathbf{B}^0\, \varphi' \bullet N_1'$, con $\varphi \rightrightarrows \varphi'$ y $N_1 \rightrightarrows N_1'$. Luego,

$$\begin{array}{lcl} M & = & \mathbf{B}^0\, \varphi \bullet N_1 \bullet N_2 \\ & \underset{\rightrightarrows \text{CaseDot}}{\rightrightarrows} & \mathbf{B}^0\, \varphi' \bullet N_1' \bullet N_2' \\ & = & M_2 \\ & \underset{\rightrightarrows \text{B-Dist}}{\rightrightarrows} & (\varphi' \bullet N_2') \bullet (N_1'\, N_2') \\ & \leftrightarrows & M_1 \end{array}$$

La última reducción se obtiene aplicando las reglas $\rightrightarrows$App y $\rightrightarrows$CaseDot sucesivas veces a partir de las hipótesis. Entonces, $M_3 = (\varphi' \bullet N_2') \bullet (N_1'\, N_2')$.

[4] Junto con diversas aplicaciones de $\rightrightarrows$R y $\rightrightarrows$CaseDot.

- $M = \vec{\theta}.\mathbf{A}^{n+1}\,\varphi\,N \rightrightarrows \vec{\theta'}.\mathbf{A}^{n}\,(\varphi \bullet N) = M_1$ (si $\vec{\theta} \rightrightarrows \vec{\theta'}$) ($\rightrightarrows$A-Dec)

La única regla a tener en cuenta para $M \rightrightarrows M_2$ es $\rightrightarrows$App. Supongamos, entonces, que $N \rightrightarrows N'$ y $\vec{\theta}.\mathbf{A}^{n+1}\,\varphi \rightrightarrows P$. Tal P sólo puede ser de la forma $P = \vec{\theta''}.\mathbf{A}^{n+1}\,\varphi'$, donde $\vec{\theta} \rightrightarrows \vec{\theta''}$ y $\varphi \rightrightarrows \varphi'$. Por hipótesis inductiva, debe existir $\vec{\theta'''}$ tal que $\vec{\theta'} \rightrightarrows \vec{\theta'''} \lhd \vec{\theta''}$. De esta manera,

$$\begin{array}{rcl} M & = & \vec{\theta}.\mathbf{A}^{n+1}\,\varphi\,N \\ & \overset{\rightrightarrows \textsc{App}}{\rightrightarrows} & M_2 \\ & = & \vec{\theta''}.\mathbf{A}^{n+1}\,\varphi'\,N' \\ & \overset{\rightrightarrows \text{A-Dec}}{\rightrightarrows} & \vec{\theta'''}.\mathbf{A}^{n}\,(\varphi' \bullet N') \\ & \lhd & M_1 \end{array}$$

Esta última reducción se deriva a partir de las hipótesis previas, aplicando las reglas $\rightrightarrows$CaseDot, $\rightrightarrows$ACase y, reiteradas veces, $\rightrightarrows$CaseTerm. Luego, $M_3 = \vec{\theta'''}.\mathbf{A}^{n}\,(\varphi' \bullet N')$.

- $M = \mathbf{B}^{n+1}\,\varphi \bullet N \rightrightarrows \mathbf{B}^{n}\,(\varphi \bullet N) = M_1$ ($\rightrightarrows$B-Dec)

El razonamiento a seguir es muy similar a lo desarrollado para el caso de $\rightrightarrows$B-Dist.

- $M = \{\!|\, c_i \mapsto N_i \,|\!\}_{i=1}^{n} \rightrightarrows \{\!|\, c_i \mapsto N_i' \,|\!\}_{i=1}^{n} = M_1$ (si $N_i \rightrightarrows N_i'$ para cada i) ($\rightrightarrows$Case)

Si $M \rightrightarrows M_2$, debe ocurrir que los términos N_i ($1 \leq i \leq n$) reduzcan a respectivos términos N_i''. Por ende, $M \overset{\rightrightarrows \textsc{Case}}{\rightrightarrows} M_2$, y $M_2 = \{\!|\, c_i \mapsto N_i'' \,|\!\}_{i=1}^{n}$.

Por hipótesis inductiva, para cada $i = 1, \cdots, n$ debe existir un término N_i''' tal que $N_i' \rightrightarrows N_i''' \lhd N_i''$. En consecuencia, basta tomar $M_3 = \{\!|\, c_i \mapsto N_i''' \,|\!\}_{i=1}^{n}$.

- $M = \mathbf{A}^{n}\theta \rightrightarrows \mathbf{A}^{n}\theta' = M_1$ (si $\theta \rightrightarrows \theta'$) ($\rightrightarrows$ACase)

Sólo puede ocurrir que $M_2 = \mathbf{A}^{n}\phi$, con ϕ tal que $\theta \rightrightarrows \phi$. El resultado sigue en forma inmediata aplicando la hipótesis inductiva (i.e., que debe existir φ tal que $\theta' \rightrightarrows \varphi \lhd \phi$).

- $M = \mathbf{B}^{n}\theta \rightrightarrows \mathbf{B}^{n}\theta' = M_1$ (si $\theta \rightrightarrows \theta'$) ($\rightrightarrows$BCase)

Ídem caso anterior.

- $M = N_1\,N_2 \rightrightarrows N_1'\,N_2' = M_1$ (si $N_1 \rightrightarrows N_1'$ y $N_2 \rightrightarrows N_2'$) ($\rightrightarrows$App)

En este caso, para la reducción de M a M_2, se deben contemplar las reglas $\rightrightarrows$K, $\rightrightarrows$S, $\rightrightarrows$A-Dist y $\rightrightarrows$A-Dec. Para ello, basta recurrir a lo analizado previamente para cada una de ellas (en su respectiva interacción con $\rightrightarrows$App).

Por último, puede ocurrir que $M_2 = N_1''\,N_2''$, con $N_1 \rightrightarrows N_1''$ y $N_2 \rightrightarrows N_2''$. Por hipótesis inductiva, alcanza con tomar $M_3 = N_1'''\,N_2'''$, donde $N_1' \rightrightarrows N_1''' \lhd N_1''$ y $N_2' \rightrightarrows N_2''' \lhd N_2''$.

- $M = \theta.N \rightrightarrows \theta'.N' = M_1$ (si $\theta \rightrightarrows \theta'$ y $N \rightrightarrows N'$) ($\rightrightarrows$CaseTerm)

Este caso precisa tener en cuenta las reglas $\rightrightarrows$CaseCons y $\rightrightarrows$CaseApp. El desarrollo para cada una de ellas es análogo a lo dicho anteriormente en sus respectivos casos (ítem (v) para el caso de $\rightrightarrows$CaseApp).

También podría suceder que $M_2 = \phi.P$, con $\theta \rightrightarrows \phi$ y $N \rightrightarrows P$. Tal situación se resuelve de manera simple aplicando la hipótesis inductiva.

- $M = \theta \bullet N \rightrightarrows \theta' \bullet N' = M_1$ (si $\theta \rightrightarrows \theta'$ y $N \rightrightarrows N'$) ($\rightrightarrows$CaseDot)

Ídem caso anterior, pero recurriendo a $\rightrightarrows$Dot, a $\rightrightarrows$B-Dist y a $\rightrightarrows$B-Dec.

□

Como ya se explicó, resta probar ahora que cada paso de $\to_{\mathrm{CL}_{\mathcal{B}_c}}$ puede simularse en $\rightrightarrows$ y, además, que cada paso de $\rightrightarrows$ puede simularse en una cantidad finita de pasos de $\to_{\mathrm{CL}_{\mathcal{B}_c}}$. Este resultado sigue a continuación.

Proposición 3.4.2. $\to_{\mathrm{CL}_{\mathcal{B}_c}} \subseteq \rightrightarrows \subseteq \twoheadrightarrow_{\mathrm{CL}_{\mathcal{B}_c}}$

Demostración La primera inclusión es consecuencia inmediata de que cada regla (y cada esquema) de $\mathrm{CL}_{\mathcal{B}_c}$ (Figura 3.1) aparece entre las reglas de $\rightrightarrows$ (Figura 3.4). Observar que, por $\rightrightarrows$R, cada vector de case binders $\vec{\theta}$ es tal que $\vec{\theta} \rightrightarrows \vec{\theta}$, lo cual permite simular cada uno de los esquemas K, S, A-Dist y A-Dec en $\rightrightarrows$ por medio de sus respectivas reglas $\rightrightarrows$K, $\rightrightarrows$S, $\rightrightarrows$A-Dist y $\rightrightarrows$A-Dec.

Para la segunda inclusión, sean $M, N \in \mathbf{CL}_{\mathcal{B}_\mathbf{c}}$ tales que $M \rightrightarrows N$. Probaremos por inducción en $M \rightrightarrows N$ que $M \twoheadrightarrow_{\mathrm{CL}_{\mathcal{B}_c}} N$.

- $M \rightrightarrows M = N$ ($\rightrightarrows$R)

$M = N$ implica que $M \twoheadrightarrow_{\mathrm{CL}_{\mathcal{B}_c}} N$ en cero pasos.

- $M = \vec{\theta}.(\vec{\phi}.\mathbf{K}\, M_1)\, M_2 \rightrightarrows \vec{\theta}'.\vec{\phi}'.M_1 = N$ (si $\vec{\theta} \rightrightarrows \vec{\theta}'$ y $\vec{\phi} \rightrightarrows \vec{\phi}'$) ($\rightrightarrows$K)

Inmediato si $|\vec{\theta}| = |\vec{\phi}| = 0$ pues, por K, $M \twoheadrightarrow_{\mathrm{CL}_{\mathcal{B}_c}} N$ en exactamente un paso. En otro caso,

$$\begin{array}{rcl} M & = & \vec{\theta}.(\vec{\phi}.\mathbf{K}\, M_1)\, M_2 \\ & \to_{\mathrm{CL}_{\mathcal{B}_c}} & \vec{\theta}.\vec{\phi}.M_1 \\ & \overset{\mathrm{HI}}{\twoheadrightarrow}_{\mathrm{CL}_{\mathcal{B}_c}} & \vec{\theta}.\vec{\phi}'.M_1 \\ & \overset{\mathrm{HI}}{\twoheadrightarrow}_{\mathrm{CL}_{\mathcal{B}_c}} & \vec{\theta}'.\vec{\phi}'.M_1 \\ & = & N \end{array}$$

- $M = \vec{\theta}.(\vec{\phi}.(\vec{\omega}.\mathbf{S}\, M_1)\, M_2)\, M_3 \rightrightarrows \vec{\theta}'.\vec{\phi}'.\vec{\omega}'.(M_1\, M_3\, (M_2\, M_3)) = N$ (si $\vec{\theta} \rightrightarrows \vec{\theta}'$, $\vec{\phi} \rightrightarrows \vec{\phi}'$ y $\vec{\omega} \rightrightarrows \vec{\omega}'$) ($\rightrightarrows$S)

Ídem caso anterior.

- $M = \vec{\theta}.(\vec{\phi}.\mathbf{A}^0\, \varphi\, M_1)\, M_2 \rightrightarrows \vec{\theta}'.\vec{\phi}'.((\varphi \bullet M_2).(M_1\, M_2)) = N$ (si $\vec{\theta} \rightrightarrows \vec{\theta}'$ y $\vec{\phi} \rightrightarrows \vec{\phi}'$) ($\rightrightarrows$A-Dist)

Ídem caso anterior.

- $M = \vec{\theta}.\mathbf{A}^{n+1}\, \varphi\, M_1 \rightrightarrows \vec{\theta}'.\mathbf{A}^n\, (\varphi \bullet M_1) = N$ (si $\vec{\theta} \rightrightarrows \vec{\theta}'$) ($\rightrightarrows$A-Dec)

Ídem caso anterior.

- $M = \theta.c \rightrightarrows N$ (si $c \mapsto N \in \theta$) ($\rightrightarrows$CaseCons)

Inmediato dado que, por (CaseCons), $M \twoheadrightarrow_{\mathrm{CL}_{\mathcal{B}_c}} N$ en exactamente un paso.

- $M = \theta.(M_1\, M_2) \rightrightarrows \theta.M_1\, M_2 = N$ ($\rightrightarrows$CaseApp)

Ídem caso anterior.

- $M = \{\!|\, c_i \mapsto M_i \,|\!\}_{i=1}^n \bullet P \rightrightarrows \{\!|\, c_i \mapsto M_i P \,|\!\}_{i=1}^n = N$ ($\rightrightarrows$DotCase)

Ídem caso anterior.

- $M = \mathbf{B}^0\, \varphi \bullet M_1 \bullet M_2 \rightrightarrows (\varphi \bullet M_2) \bullet (M_1\, M_2) = N$ ($\rightrightarrows$B-Dist)

Ídem caso anterior.

- $M = \mathbf{B}^{n+1}\, \varphi \bullet M_1 \rightrightarrows \mathbf{B}^n\, (\varphi \bullet M_1) = N$ ($\rightrightarrows$B-Dec)

Ídem caso anterior.

- $M = \{\!|\, c_i \mapsto M_i \,|\!\}_{i=1}^n \rightrightarrows \{\!|\, c_i \mapsto M'_i \,|\!\}_{i=1}^n = N$ (si $M_i \rightrightarrows M'_i$ para cada i) ($\rightrightarrows$Case)

$$\begin{array}{rcl} M & = & \{\!|\, c_1 \mapsto M_1; \cdots; c_n \mapsto M_n \,|\!\} \\ & \overset{\mathrm{HI}}{\twoheadrightarrow_{\mathrm{CL}_{\mathcal{B}_c}}} & \{\!|\, c_1 \mapsto M'_1; \cdots; c_n \mapsto M_n \,|\!\} \\ & \vdots & \\ & \overset{\mathrm{HI}}{\twoheadrightarrow_{\mathrm{CL}_{\mathcal{B}_c}}} & \{\!|\, c_1 \mapsto M'_1; \cdots; c_n \mapsto M'_n \,|\!\} \\ & = & N \end{array}$$

- $M = \mathbf{A}^n\theta \rightrightarrows \mathbf{A}^n\theta' = N$ (si $\theta \rightrightarrows \theta'$) ($\rightrightarrows$ACase)

Inmediato a partir de la hipótesis inductiva para θ.

- $M = \mathbf{B}^n\theta \rightrightarrows \mathbf{B}^n\theta' = N$ (si $\theta \rightrightarrows \theta'$) ($\rightrightarrows$BCase)

Ídem caso anterior.

- $M = M_1\, M_2 \rightrightarrows M'_1\, M'_2 = N$ (si $M_1 \rightrightarrows M'_1$ y $M_2 \rightrightarrows M'_2$) ($\rightrightarrows$App)

$$\begin{array}{rcl} M & = & M_1\, M_2 \\ & \overset{\mathrm{HI}}{\twoheadrightarrow_{\mathrm{CL}_{\mathcal{B}_c}}} & M'_1\, M_2 \\ & \overset{\mathrm{HI}}{\twoheadrightarrow_{\mathrm{CL}_{\mathcal{B}_c}}} & M'_1\, M'_2 \\ & = & N \end{array}$$

- $M = \theta.M_1 \rightrightarrows \theta'.M'_1 = N$ (si $\theta \rightrightarrows \theta'$ y $M_1 \rightrightarrows M'_1$) ($\rightrightarrows$CaseTerm)

Ídem caso anterior.

- $M = \theta \bullet M_1 \rightrightarrows \theta' \bullet M'_1 = N$ (si $\theta \rightrightarrows \theta'$ y $M_1 \rightrightarrows M'_1$) ($\rightrightarrows$CaseDot)

Ídem caso anterior.

□

A partir de las proposiciones 3.4.1 y 3.4.2, el método de Tait-Martin-Löf permite concluir que $\rightarrow_{\mathrm{CL}_{\mathcal{B}_c}}$ es una relación confluente. De esta manera, queda demostrada la siguiente Proposición:

Proposición 3.4.3. $\rightarrow_{\mathrm{CL}_{\mathcal{B}_c}}$ es confluente.

3.5. Representación de la abstracción en $\mathrm{CL}_{\mathcal{B}_c}$

Vamos a mostrar que con nuestro sistema de combinadores podremos expresar la noción de abstracción, esto es, simular el tener variables ligadas que resultarán sustituidas en el cuerpo de la misma, como pasa en $\lambda\mathcal{B}_\mathcal{C}$, pero ahora con una sintaxis sin ligadores. Esto permite que la sintaxis de $\mathrm{CL}_{\mathcal{B}_c}$ sea más cercana a una formulación de primer orden, aunque sin serlo de todas formas[5].

3.5.1. El operador λ^*

(Id)	$\lambda^* x.x$	$=$	$\mathbf{S}\,\mathbf{K}\,\mathbf{K}$	
(K-Intro)	$\lambda^* x.M$	$=$	$\mathbf{K}\,M$	$(x \notin \mathrm{FV}(M))$
(Const)	$\lambda^* x.M\,x$	$=$	M	$(x \notin \mathrm{FV}(M))$
(S-Intro)	$\lambda^* x.M\,N$	$=$	$\mathbf{S}\,(\lambda^* x.M)\,(\lambda^* x.N)$	$(x \in \mathrm{FV}(MN), N \neq x)$
(A-Intro)	$\lambda^* x.\theta.M$	$=$	$\mathbf{A}^0\,(\lambda^* x.\theta)\,(\lambda^* x.M)$	
(A-Inc)	$\lambda^* x.\mathbf{A}^n\,\theta$	$=$	$\mathbf{A}^{n+1}\,(\lambda^* x.\theta)$	
(CaseDist)	$\lambda^* x.\{\!\|\, c_i \mapsto M_i \,\|\!\}_{i=1}^n$	$=$	$\{\!\|\, c_i \mapsto \lambda^* x.M_i \,\|\!\}_{i=1}^n$	
(B-Intro)	$\lambda^* x.\theta \bullet M$	$=$	$\mathbf{B}^0\,(\lambda^* x.\theta) \bullet (\lambda^* x.M)$	
(B-Inc)	$\lambda^* x.\mathbf{B}^n\,\theta$	$=$	$\mathbf{B}^{n+1}\,(\lambda^* x.\theta)$	

Figura 3.5: El operador λ^*

En la Figura 3.5 mostramos las reglas de conversión que definen nuestro operador de abstracción. Nótese que esto debe hacerse tanto para los términos como para los case binders, puesto que unos a otros se utilizan en forma cruzada. Por otro lado, observar también que esta definición extiende una versión clásica de λ^*.

Hacemos a continuación una explicación del funcionamiento general de estos combinadores en función de la necesidad de definir el operador de abstracción, la que conduce a la inclusión de aquéllos.

En primer lugar digamos que así como se aplica un término a otro, estamos creando la posibilidad de aplicar un case binder a un término, denotada con $\theta \bullet M$, lo que devendrá en la posibilidad de sustitución una vez ejecutada la β-reducción correspondiente (para el caso en que a su vez θ represente una abstracción). El combinador $\mathbf{A}^0$ se utiliza entonces para descomponer la posibilidad de aplicación tanto en un término de la forma $\theta.M$ para el término M como para el case binder θ. En este caso, se intenta aplicar tanto uno como otro, al argumento, pero cada uno con la aplicación asociada a su sort. Pero, a su vez, $\mathbf{A}^0$ podría aparecer otra vez bajo el alcance del abstractor, con lo cual hará falta otro combinador. Se introduce entonces $\mathbf{A}^1$, de manera de manejar adecuadamente el anterior. Y así sucesivamente. Entonces se hace necesario contar con una cantidad infinita enumerable de ellos. Análogamente, también hará falta definir la abstracción sobre el caso de la aplicación con la que contamos, $\theta \bullet M$, dando lugar a la definición de los combinadores $\mathbf{B}^n$. Este sistema

[5]Es claro que $\mathrm{CL}_{\mathcal{B}_c}$ no puede ser un sistema de reescritura de términos de primer orden debido a que los case binders siguen siendo conjuntos tal como lo eran en $\lambda\mathcal{B}_\mathcal{C}$. Creemos que no hay una formulación de primer orden fácil o natural, y buscarla nos alejaría del objetivo de mantener las características esenciales de $\lambda\mathcal{B}_\mathcal{C}$.

pues, creemos, representa aquél que con mínima sintaxis y reglas adicionales, permite tal mecanismo.

Al combinador $\mathbf{A}^n$ se lo puede ver como el que toma el siguiente argumento y lo convierte en aplicación por debajo de él ($n > 0$), o finalmente toma los dos siguientes argumentos y los convierte en aplicación con entorno ($n = 0$), tal como funciona el combinador $\mathbf{S}$ en la lógica combinatoria clásica. Lo mismo puede leerse en los combinadores $\mathbf{B}^n$, pero ahora para operar con case binders.

3.5.1.1. Propiedades de λ^*

En primer lugar, se tiene el siguiente resultado que indica que el operador λ^* permite deshacerse del concepto de variable ligada y del alcance de las variables, lo cual constituyó la motivación inicial de su definición.

Proposición 3.5.1. Sea $x \in \mathcal{V}$. Para todo término M (resp. case binder θ), se tiene que $x \notin \mathrm{FV}(\lambda^* x.M)$ (resp. $x \notin \mathrm{FV}(\lambda^* x.\theta)$)

Demostración Por inducción en M (resp. θ). □

Ahora bien, veamos la propiedad que indica que tanto para los términos como para los case binders, λ^* resulta en una abstracción en el sentido descripto anteriormente.

Proposición 3.5.2. Sean $M, N \in \mathbf{CL}_{\mathcal{B}_\mathbf{c}}$. Entonces,

$$\begin{array}{rcl} M \in \mathbf{CL}^{\mathrm{T}}_{\mathcal{B}_\mathbf{c}} & \Rightarrow & (\lambda^* x.M)\, N \twoheadrightarrow_{\mathrm{CL}_{\mathcal{B}_c}} M[x \,/\, N] \\ M = \theta \in \mathbf{CL}^{\mathrm{CB}}_{\mathcal{B}_\mathbf{c}} & \Rightarrow & (\lambda^* x.\theta) \bullet N \twoheadrightarrow_{\mathrm{CL}_{\mathcal{B}_c}} \theta[x \,/\, N] \end{array}$$

Demostración Por inducción en M. Dado que $M \in \mathbf{CL}_{\mathcal{B}_\mathbf{c}}$, la inducción se divide en dos partes; cada una de ellas correspondiendo al sort de los términos y al sort de los case binders respectivamente.

En primera instancia, supongamos que $M \in \mathbf{CL}^{\mathrm{T}}_{\mathcal{B}_\mathbf{c}}$.

- $M = a \in \mathcal{A}$

Si $a \in \mathcal{C}$ o si $a = \mathbf{K}, \mathbf{S}$, se tiene que $\mathrm{FV}(a) = \emptyset$. Luego,

$$\begin{array}{rcl} (\lambda^* x.M)\, N & = & (\lambda^* x.a)\, N \\ & \overset{\text{K-Intro}}{=} & (\mathbf{K}\, a)\, N \\ & \to^{\mathrm{K}}_{\mathrm{CL}_{\mathcal{B}_c}} & a \\ & \overset{\text{Sub-Atom}}{=} & a[x \,/\, N] \\ & = & M[x \,/\, N] \end{array}$$

Por otra parte, puede ocurrir que $a = y \in \mathcal{V}$. Si $y \neq x$, el desarrollo es análogo a lo anterior (pues $x \notin \mathrm{FV}(y) = \{y\}$). Si, en cambio, $y = x$, sucede lo siguiente:

$$\begin{array}{rcl} (\lambda^* x.M)\, N & = & (\lambda^* x.x)\, N \\ & \overset{\text{Id}}{=} & (\mathbf{S}\, \mathbf{K}\, \mathbf{K})\, N \\ & \to^{\mathrm{S}}_{\mathrm{CL}_{\mathcal{B}_c}} & \mathbf{K}\, N\, (\mathbf{K}\, N) \\ & \to^{\mathrm{K}}_{\mathrm{CL}_{\mathcal{B}_c}} & N \\ & \overset{\text{Sub-Atom}}{=} & x[x \,/\, N] \\ & = & M[x \,/\, N] \end{array}$$

- $M = M_1\, M_2$

Si $M_2 = x$ y además $x \notin \mathrm{FV}(M_1)$,

$$\begin{array}{rcl}(\lambda^* x.M)\, N & = & (\lambda^* x.M_1\, x)\, N \\ & \overset{\textsc{Const}}{=} & M_1\, N \\ & \overset{\textsc{Sub-Atom}}{=} & M_1[x \,/\, N]\, x[x \,/\, N] \\ & \overset{\textsc{Sub-App}}{=} & (M_1\, x)[x \,/\, N] \\ & = & M[x \,/\, N]\end{array}$$

En cualquier otro caso,

$$\begin{array}{rcl}(\lambda^* x.M)\, N & = & (\lambda^* x.M_1\, M_2)\, N \\ & \overset{\textsc{S-Intro}}{=} & (\mathbf{S}\, (\lambda^* x.M_1)\, (\lambda^* x.M_2))\, N \\ & \to^{\mathrm{S}}_{\mathrm{CL}_{\mathcal{B}_c}} & (\lambda^* x.M_1)\, N\, ((\lambda^* x.M_2)\, N) \\ & \twoheadrightarrow^{\mathrm{HI}}_{\mathrm{CL}_{\mathcal{B}_c}} & M_1[x \,/\, N]\, M_2[x \,/\, N] \\ & \overset{\textsc{Sub-App}}{=} & (M_1\, M_2)[x \,/\, N] \\ & = & M[x \,/\, N]\end{array}$$

- $M = \theta.M_1$

$$\begin{array}{rcl}(\lambda^* x.M)\, N & = & (\lambda^* x.\theta.M_1)\, N \\ & \overset{\textsc{A-Intro}}{=} & (\mathbf{A}^0\, (\lambda^* x.\theta)\, (\lambda^* x.M_1))\, N \\ & \to^{\textsc{A-Dist}}_{\mathrm{CL}_{\mathcal{B}_c}} & ((\lambda^* x.\theta) \bullet N).((\lambda^* x.M_1)\, N) \\ & \twoheadrightarrow^{\mathrm{HI}}_{\mathrm{CL}_{\mathcal{B}_c}} & \theta[x \,/\, N].M_1[x \,/\, N] \\ & \overset{\textsc{Sub-CaseTerm}}{=} & (\theta.M_1)[x \,/\, N] \\ & = & M[x \,/\, N]\end{array}$$

- $M = \mathbf{A}^n\, \theta$ $(n \in \mathbb{N})$

$$\begin{array}{rcl}(\lambda^* x.M)\, N & = & (\lambda^* x.\mathbf{A}^n\, \theta)\, N \\ & \overset{\textsc{A-Inc}}{=} & (\mathbf{A}^{n+1}\, (\lambda^* x.\theta))\, N \\ & \to^{\textsc{A-Dec}}_{\mathrm{CL}_{\mathcal{B}_c}} & \mathbf{A}^n((\lambda^* x.\theta) \bullet N) \\ & \twoheadrightarrow^{\mathrm{HI}}_{\mathrm{CL}_{\mathcal{B}_c}} & \mathbf{A}^n\theta[x \,/\, N] \\ & \overset{\textsc{Sub-A}}{=} & (\mathbf{A}^n\, \theta)[x \,/\, N] \\ & = & M[x \,/\, N]\end{array}$$

Por otro lado, supongamos ahora que $M = \theta \in \mathbf{CL}^{\mathrm{CB}}_{\mathcal{B}_\mathbf{c}}$. Luego, la inducción sobre θ prosigue de la siguiente manera:

- $\theta = \{\!|\, c_1 \mapsto M_1; \cdots ; c_n \mapsto M_n \,|\!\}$ $(n > 0)$

$$\begin{array}{rcl}
(\lambda^* x.\theta) \bullet N & = & (\lambda^* x.\{\!|\, c_1 \mapsto M_1; \cdots ; c_n \mapsto M_n \,|\!\}) \bullet N \\
& \overset{\text{CaseDist}}{=} & (\{\!|\, c_1 \mapsto \lambda^* x.M_1; \cdots ; c_n \mapsto \lambda^* x.M_n \,|\!\}) \bullet N \\
& \overset{\text{DotCase}}{\to_{\text{CL}_{\mathcal{B}_c}}} & \{\!|\, c_1 \mapsto (\lambda^* x.M_1)\, N; \cdots ; c_n \mapsto (\lambda^* x.M_n)\, N \,|\!\} \\
& \overset{\text{HI}}{\twoheadrightarrow_{\text{CL}_{\mathcal{B}_c}}} & \{\!|\, c_1 \mapsto M_1[x \,/\, N]; \cdots ; c_n \mapsto M_n[x \,/\, N] \,|\!\} \\
& \overset{\text{Sub-Case}}{=} & (\{\!|\, c_1 \mapsto M_1; \cdots ; c_n \mapsto M_n \,|\!\})[x \,/\, N] \\
& = & \theta[x \,/\, N]
\end{array}$$

- $\theta = \varphi \bullet M_1$

$$\begin{array}{rcl}
(\lambda^* x.\theta) \bullet N & = & (\lambda^* x.\varphi \bullet M_1) \bullet N \\
& \overset{\text{B-Intro}}{=} & (\mathbf{B}^0(\lambda^* x.\varphi) \bullet (\lambda^* x.M_1)) \bullet N \\
& \overset{\text{B-Dist}}{\to_{\text{CL}_{\mathcal{B}_c}}} & ((\lambda^* x.\varphi) \bullet N) \bullet ((\lambda^* x.M_1)\, N) \\
& \overset{\text{HI}}{\twoheadrightarrow_{\text{CL}_{\mathcal{B}_c}}} & \varphi[x \,/\, N] \bullet M_1[x \,/\, N] \\
& \overset{\text{Sub-Dot}}{=} & (\varphi \bullet M_1)[x \,/\, N] \\
& = & \theta[x \,/\, N]
\end{array}$$

- $\theta = \mathbf{B}^n\, \varphi$ $(n \in \mathbb{N})$

$$\begin{array}{rcl}
(\lambda^* x.\theta) \bullet N & = & (\lambda^* x.\mathbf{B}^n\, \varphi) \bullet N \\
& \overset{\text{B-Inc}}{=} & (\mathbf{B}^{n+1}(\lambda^* x.\varphi)) \bullet N \\
& \overset{\text{B-Dec}}{\to_{\text{CL}_{\mathcal{B}_c}}} & \mathbf{B}^n((\lambda^* x.\varphi) \bullet N) \\
& \overset{\text{HI}}{\twoheadrightarrow_{\text{CL}_{\mathcal{B}_c}}} & \mathbf{B}^n \varphi[x \,/\, N] \\
& \overset{\text{Sub-B}}{=} & (\mathbf{B}^n\, \varphi)[x \,/\, N] \\
& = & \theta[x \,/\, N]
\end{array}$$

□

3.6. Traducciones entre $\lambda\mathcal{B}_c$ y $\mathbf{CL}_{\mathcal{B}_c}$

En esta sección presentaremos traducciones entre estos dos formalismos. La conversión de $\lambda\mathcal{B}_c$ a $\text{CL}_{\mathcal{B}_c}$ (sección 3.6.1) se construye a partir de la definición de λ^*, estudiada anteriormente. La conversión inversa (sección 3.6.2) requiere la consideración adicional de los meta-operadores $\circ$ y $\underline{\lambda}$, presentados en la Definición 3.3.6. Para este caso, se probará que la traducción se comporta de la manera esperada. En la sección 3.6.3 enunciaremos dos proposiciones que permiten ver a estas traducciones como pseudo inversas.

En cualquier caso, las traducciones dadas pueden verse como extensiones a las respectivas traducciones clásicas entre el λ-cálculo y la lógica combinatoria.

3.6.1. De $\lambda\mathcal{B}_c$ a $\mathbf{CL}_{\mathcal{B}_c}$

La Figura 3.6 muestra la definición inductiva de la función de traducción $(\bullet)_{\mathrm{CL}}$. Es de especial interés el caso de la abstracción, que se traduce mediante el operador λ^*.

(C-Var)	x_{CL}	$=$	x	$(x \in \mathcal{V})$
(C-Cons)	c_{CL}	$=$	c	$(c \in \mathcal{C})$
(C-App)	$(M\,N)_{\mathrm{CL}}$	$=$	$M_{\mathrm{CL}}\,N_{\mathrm{CL}}$	
(C-Abs)	$(\lambda x.M)_{\mathrm{CL}}$	$=$	$\lambda^* x.M_{\mathrm{CL}}$	
(C-Case)	$(\theta.M)_{\mathrm{CL}}$	$=$	$\theta_{\mathrm{CL}}.M_{\mathrm{CL}}$	
(C-CB)	$(\{\!\| c_i \mapsto M_i \|\!\}_{i=1}^n)_{\mathrm{CL}}$	$=$	$\{\!\| c_i \mapsto M_{i\mathrm{CL}} \|\!\}_{i=1}^n$	

Figura 3.6: Conversión de $\lambda\mathcal{B}_c$ a $\mathrm{CL}_{\mathcal{B}_c}$

3.6.2. De $\mathbf{CL}_{\mathcal{B}_c}$ a $\lambda\mathcal{B}_c$

La definición de esta conversión, notada $(\bullet)_\lambda$, aparece en la Figura 3.7. Asumiremos que las distintas variables ligadas introducidas por las ecuaciones son variables frescas.

(L-Var)	x_λ	$=$	x	$(x \in \mathcal{V})$
(L-Cons)	c_λ	$=$	c	$(c \in \mathcal{C})$
(L-K)	$\mathbf{K}_\lambda$	$=$	$\lambda xy.x$	
(L-S)	$\mathbf{S}_\lambda$	$=$	$\lambda xyz.x\,z\,(y\,z)$	
(L-App)	$(M\,N)_\lambda$	$=$	$M_\lambda\,N_\lambda$	
(L-A)	$(\mathbf{A}^n\,\theta)_\lambda$	$=$	$\lambda x_1 \cdots x_{n+2}.$ $(\theta_\lambda \circ x_1 \circ \cdots \circ x_n \circ x_{n+2}).(x_{n+1}\,x_{n+2})$	
(L-Case)	$(\theta.M)_\lambda$	$=$	$\theta_\lambda.M_\lambda$	
(L-CB)	$(\{\!\| c_i \mapsto M_i \|\!\}_{i=1}^n)_\lambda$	$=$	$\{\!\| c_i \mapsto M_{i\lambda} \|\!\}_{i=1}^n$	
(L-Dot)	$(\theta \bullet M)_\lambda$	$=$	$\theta_\lambda \circ M_\lambda$	
(L-B)	$(\mathbf{B}^n\,\theta)_\lambda$	$=$	$\underline{\lambda} x_1 \cdots x_{n+2}.$ $\theta_\lambda \circ x_1 \circ \cdots \circ x_n \circ x_{n+2} \circ (x_{n+1}\,x_{n+2})$	

Figura 3.7: Conversión de $\mathrm{CL}_{\mathcal{B}_c}$ a $\lambda\mathcal{B}_c$

La Proposición 3.6.1 garantiza que un paso de reducción en $\mathrm{CL}_{\mathcal{B}_c}$ se simula correctamente al traducir los términos involucrados a $\lambda\mathcal{B}_c$. En primera instancia, es preciso realizar una serie de observaciones que serán de utilidad a lo largo de la prueba. La primera de ellas hace referencia a que el mapeo, en efecto, tiene como codominio el conjunto de términos $\Lambda\mathcal{B}_c$ [6]. En especial, se nota que un case binder de $\mathrm{CL}_{\mathcal{B}_c}$cualquiera, al ser traducido, arroja un case binder clásico. Esta última afirmación será utilizada en forma implícita durante el resto de la presente sección, excepto en determinados pasajes donde sí requiera una mención explícita.

[6] Esto, a simple vista, podría no ser obvio debido a la utilización de los meta-operadores $\circ$ y $\underline{\lambda}$ en las ecuaciones.

Observación 3.6.1. Sea $M \in \mathbf{CL}^{\mathrm{T}}_{\mathcal{B}_\mathbf{c}}$. Luego, $M_\lambda \in \Lambda\mathcal{B}_c$ si M es un término, o bien, de lo contrario, es un case binder clásico de $\lambda\mathcal{B}_c$ (i.e., existe $n \geq 0$, constructores $c_1, \cdots, c_n$ y términos $M_1, \cdots, M_n$ tales que $\theta_\lambda = \{\!|\, c_i \mapsto M_i \,|\!\}_{i=1}^n$).

Demostración Por inducción, y teniendo en cuenta que los operadores $\circ$ y $\underline{\lambda}$ sólo definen transformaciones sobre case binders de $\lambda\mathcal{B}_c$. □

La próxima observación extiende la regla CaseLam de $\lambda\mathcal{B}_c$ a los casos donde se tiene un vector de case binders (de $\lambda\mathcal{B}_c$) en lugar de un único case binder.

Observación 3.6.2. Sea $M \in \Lambda\mathcal{B}_c$ y $\vec{\theta}$ un vector de case binders. Luego,

$$\vec{\theta}.\lambda x.M \twoheadrightarrow \lambda x.\vec{\theta}.M$$

Demostración Por inducción en $\vec{\theta}$. Si $\vec{\theta} = \emptyset$, vale la igualdad. En otro caso, el resultado es inmediato a partir de la hipótesis inductiva y de la aplicación de la regla CaseLam. □

Una última observación trata sobre reducciones en los case binders resultantes de aplicar los meta-operadores $\circ$ y $\underline{\lambda}$.

Observación 3.6.3. Sean $\theta, \theta' \in \mathcal{B}$ tales que $\theta \to \theta'$. Luego, dado $M \in \Lambda\mathcal{B}_c$ y $x \in \mathcal{V}$,

$$\begin{array}{rcl} \theta \circ M & \to & \theta' \circ M \\ \underline{\lambda}x.\theta & \to & \underline{\lambda}x.\theta' \end{array}$$

Demostración Dado que $\theta = \{\!|\, c_i \mapsto M_i \,|\!\}_{i=1}^n$ para cierto $n \geq 0$, constructores $c_1, \cdots, c_n$ y términos $M_1, \cdots, M_n$, la reducción debe darse en algún M_j $(1 \leq j \leq n)$, i.e., $M_j \to M'_j$ y $\theta' = \{\!|\, c_1 \mapsto M_1; \cdots; c_j \mapsto M'_j; \cdots; c_n \mapsto M_n \,|\!\}$. Entonces,

$$\begin{array}{rcl} \theta \circ M & = & \{\!|\, c_1 \mapsto M_1 M; \cdots; c_j \mapsto M_j M; \cdots; c_n \mapsto M_n M \,|\!\} \\ & \to & \{\!|\, c_1 \mapsto M_1 M; \cdots; c_j \mapsto M'_j M; \cdots; c_n \mapsto M_n M \,|\!\} \\ & = & \theta' \circ M \end{array}$$

Para la otra parte, el razonamiento es similar. □

De esta manera, ya estamos listos para probar el resultado que nos proponíamos:

Proposición 3.6.1. Sean $M, N \in \mathbf{CL}_{\mathcal{B}_\mathbf{c}}$ tales que $M \to_{\mathrm{CL}_{\mathcal{B}_c}} N$. Entonces,

$$M_\lambda \twoheadrightarrow_{\lambda\mathcal{B}_c} N_\lambda$$

Demostración Por inducción en M. Al ser $M \in \mathbf{CL}_{\mathcal{B}_\mathbf{c}}$, la inducción debe separarse en dos partes: la primera de ellas correspondiendo al sort de los términos y, la restante, al sort de los case binders. Por simplicidad, en cada caso analizado, omitiremos el cálculo detallado de M_λ y N_λ. No obstante, es inmediato aplicando las distintas ecuaciones de la Figura 3.7 según corresponda en cada caso. Además, la notación $\vec{\theta}_\lambda$ indica, como es de esperar, la aplicación de la traducción a cada componente de $\vec{\theta}$.

Supongamos primero que $M \in \mathbf{CL}^{\mathrm{T}}_{\mathcal{B}_\mathbf{c}}$.

- $M = a \in \mathcal{A}$

a es $\mathrm{CL}_{\mathcal{B}_c}$-forma normal, de manera que la implicación es vacuamente cierta.

- $M = M_1\, M_2$

Para este caso, es necesario tener en cuenta varias alternativas posibles.

(i) $M = \vec{\theta}.(\vec{\phi}.\mathbf{K}\, P)\, M_2 \overset{\text{K}}{\to} \vec{\theta}.\vec{\phi}.P = N$

$$\begin{array}{rcl} M_\lambda & = & (\vec{\theta}.(\vec{\phi}.\mathbf{K}\, P)\, M_2))_\lambda \\ & = & \vec{\theta}_\lambda.(\vec{\phi}_\lambda.(\lambda xy.x)\, P_\lambda)\, M_{2\lambda} \\ & \overset{\text{Obs. 3.6.2}}{\twoheadrightarrow} & \vec{\theta}_\lambda.((\lambda xy.\vec{\phi}_\lambda.x)\, P_\lambda)\, M_{2\lambda} \\ & \overset{\textsc{AppLam}}{\to} & \vec{\theta}_\lambda.(\lambda y.\vec{\phi}_\lambda.P_\lambda)\, M_{2\lambda} \\ & \overset{\text{Obs. 3.6.2}}{\twoheadrightarrow} & (\lambda y.\vec{\theta}_\lambda.\vec{\phi}_\lambda.P_\lambda)\, M_{2\lambda} \\ & \overset{\textsc{AppLam}}{\to} & \vec{\theta}_\lambda.\vec{\phi}_\lambda.P_\lambda \\ & = & N_\lambda \end{array}$$

Notar que las variables x e y son frescas, de manera que no ocurren en los vectores de case binders de M ni en P_λ.

(ii) $M = \vec{\theta}.(\vec{\phi}.(\vec{\omega}.\mathbf{S}\, P)\, Q)\, M_2 \overset{\text{S}}{\to} \vec{\theta}.\vec{\phi}.\vec{\omega}.(P\, M_2\, (Q\, M_2)) = N$

$$\begin{array}{rcl} M_\lambda & = & (\vec{\theta}.(\vec{\phi}.(\vec{\omega}.\mathbf{S}\, P)\, Q)\, M_2)_\lambda \\ & = & \vec{\theta}_\lambda.(\vec{\phi}_\lambda.(\vec{\omega}_\lambda.(\lambda xyz.x\, z\, (y\, z))\, P_\lambda)\, Q_\lambda)\, M_{2\lambda} \\ & \overset{\text{Obs. 3.6.2}}{\twoheadrightarrow} & \vec{\theta}_\lambda.(\vec{\phi}_\lambda.((\lambda xyz.\vec{\omega}_\lambda.(x\, z\, (y\, z)))\, P_\lambda)\, Q_\lambda)\, M_{2\lambda} \\ & \overset{\textsc{AppLam}}{\to} & \vec{\theta}_\lambda.(\vec{\phi}_\lambda.\lambda yz.\vec{\omega}_\lambda.(P_\lambda\, z\, (y\, z))\, Q_\lambda)\, M_{2\lambda} \\ & \overset{\text{Obs. 3.6.2}}{\twoheadrightarrow} & \vec{\theta}_\lambda.((\lambda yz.\vec{\phi}_\lambda.\vec{\omega}_\lambda.(P_\lambda\, z\, (y\, z)))\, Q_\lambda)\, M_{2\lambda} \\ & \overset{\textsc{AppLam}}{\to} & \vec{\theta}_\lambda.\lambda z.\vec{\phi}_\lambda.\vec{\omega}_\lambda.(P_\lambda\, z\, (Q_\lambda\, z))\, M_{2\lambda} \\ & \overset{\text{Obs. 3.6.2}}{\twoheadrightarrow} & (\lambda z.\vec{\theta}_\lambda.\vec{\phi}_\lambda.\vec{\omega}_\lambda.(P_\lambda\, z\, (Q_\lambda\, z)))\, M_{2\lambda} \\ & \overset{\textsc{AppLam}}{\to} & \vec{\theta}_\lambda.\vec{\phi}_\lambda.\vec{\omega}_\lambda.(P_\lambda\, M_{2\lambda}\, (Q_\lambda\, M_{2\lambda})) \\ & = & N_\lambda \end{array}$$

(iii) $M = \vec{\theta}.(\vec{\phi}.\mathbf{A}^0\, \varphi\, P)\, M_2 \overset{\text{A-Dist}}{\to} \vec{\theta}.\vec{\phi}.((\varphi \bullet M_2).(P\, M_2)) = N$

$$\begin{array}{rcl} M_\lambda & = & (\vec{\theta}.(\vec{\phi}.\mathbf{A}^0\, \varphi\, P)\, M_2)_\lambda \\ & = & \vec{\theta}_\lambda.(\vec{\phi}_\lambda.\lambda x_1x_2.(\varphi_\lambda \circ x_2).(x_1\, x_2)\, P_\lambda)\, M_{2\lambda} \\ & \overset{\text{Obs. 3.6.2}}{\twoheadrightarrow} & \vec{\theta}_\lambda.((\lambda x_1x_2.\vec{\phi}_\lambda.((\varphi_\lambda \circ x_2).(x_1\, x_2)))\, P_\lambda)\, M_{2\lambda} \\ & \overset{\textsc{AppLam}}{\to} & \vec{\theta}_\lambda.\lambda x_2.\vec{\phi}_\lambda.((\varphi_\lambda \circ x_2).(P_\lambda\, x_2))\, M_{2\lambda} \\ & \overset{\text{Obs. 3.6.2}}{\twoheadrightarrow} & (\lambda x_2.\vec{\theta}_\lambda.\vec{\phi}_\lambda.((\varphi_\lambda \circ x_2).(P_\lambda\, x_2)))\, M_{2\lambda} \\ & \overset{\textsc{AppLam}}{\to} & \vec{\theta}_\lambda.\vec{\phi}_\lambda.((\varphi_\lambda \circ M_{2\lambda}).(P_\lambda\, M_{2\lambda})) \\ & = & N_\lambda \end{array}$$

(iv) $M = \vec{\theta}.\mathbf{A}^{n+1}\,\varphi\,M_2 \overset{\text{A-Dec}}{\to} \vec{\theta}.\mathbf{A}^n\,(\varphi \bullet M_2) = N$

$$\begin{array}{rcl} M_\lambda & = & (\vec{\theta}.\mathbf{A}^{n+1}\,\varphi\,M_2)_\lambda \\ & = & \vec{\theta}_\lambda.\lambda x_1 \cdots x_{n+3}.(\varphi_\lambda \circ x_1 \circ \cdots \circ x_{n+1} \circ x_{n+3}).(x_{n+2}\,x_{n+3})\,M_{2\lambda} \\ & \overset{\text{Obs. 3.6.2}}{\twoheadrightarrow} & (\lambda x_1.\vec{\theta}_\lambda.\lambda x_2 \cdots x_{n+3}.(\varphi_\lambda \circ x_1 \circ \cdots \circ x_{n+1} \circ x_{n+3}).(x_{n+2}\,x_{n+3}))\,M_{2\lambda} \\ & \overset{\text{AppLam}}{\to} & \vec{\theta}_\lambda.\lambda x_2 \cdots x_{n+3}.(\varphi_\lambda \circ M_{2\lambda} \circ x_2 \circ \cdots \circ x_{n+1} \circ x_{n+3}).(x_{n+2}\,x_{n+3}) \\ & = & \vec{\theta}_\lambda.\lambda x_1 \cdots x_{n+2}.(\varphi_\lambda \circ M_{2\lambda} \circ x_1 \circ \cdots \circ x_n \circ x_{n+2}).(x_{n+1}\,x_{n+2}) \\ & = & N_\lambda \end{array}$$

(v) Reducción interna
En tal caso, ocurrirá que $M_1 \to M_1'$ o bien $M_2 \to M_2'$. Apoyándonos en la hipótesis inductiva, el resultado es inmediato.

- $M = \theta.M_1$

Este escenario también presenta distintos subcasos posibles.

(i) $M = \theta.c \overset{\text{CaseCons}}{\to} N$ (si $c \mapsto N \in \theta$)
De ser así, $\theta = \{\!|\, c_i \mapsto N_i \,|\!\}_{i=1}^n$ y $c = c_j$ y $N = N_j$ para algún $j = 1, \cdots, n$. Luego,

$$\begin{array}{rcl} M_\lambda & = & (\theta.c)_\lambda \\ & = & \theta_\lambda.c \\ & = & \{\!|\, c_i \mapsto N_{i\lambda} \,|\!\}_{i=1}^n.c \\ & \overset{\text{CaseCons}}{\to} & N_{j\lambda} \\ & = & N_\lambda \end{array}$$

(ii) $M = \theta.(P\,Q) \overset{\text{CaseApp}}{\to} \theta.P\,Q = N$

$$\begin{array}{rcl} M_\lambda & = & (\theta.(P\,Q))_\lambda \\ & = & \theta_\lambda.(P_\lambda\,Q_\lambda) \\ & \overset{\text{CaseApp}}{\to} & \theta_\lambda.P_\lambda\,Q_\lambda \\ & = & N_\lambda \end{array}$$

(iii) Reducción interna
De forma similar a lo estudiado anteriormente, debe ocurrir que $\theta \to \theta'$ o bien $M_1 \to M_1'$. En cualquier caso, el resultado es inmediato utilizando la hipótesis inductiva.

- $M = \mathbf{A}^n\,\theta$ $(n \in \mathbb{N})$

A diferencia de los casos previos, en esta oportunidad M no puede poseer un redex en la raíz. Por ende, la reducción debe ser interna, i.e., debe existir $\theta' \in \mathbf{CL}_{\mathcal{B}_\mathbf{c}}^{\text{CB}}$ tal que $\theta \to \theta'$, y $N = \mathbf{A}^n\,\theta'$. En consecuencia,

$$\begin{array}{rcl} M_\lambda & = & (\mathbf{A}^n\theta)_\lambda \\ & = & \lambda x_1 \cdots x_{n+2}.(\theta_\lambda \circ x_1 \circ \cdots \circ x_n \circ x_{n+2}).(x_{n+1}\,x_{n+2}) \\ & \overset{\text{HI}}{\twoheadrightarrow} & \lambda x_1 \cdots x_{n+2}.(\theta'_\lambda \circ x_1 \circ \cdots \circ x_n \circ x_{n+2}).(x_{n+1}\,x_{n+2}) \\ & = & N_\lambda \end{array}$$

Notar que el paso donde se aplica la hipótesis inductiva también utiliza lo remarcado en la Observación 3.6.3: razonando en forma inductiva, a partir de ella se obtiene que

$$\theta_\lambda \circ x_1 \circ \cdots \circ x_n \circ x_{n+2} \twoheadrightarrow \theta'_\lambda \circ x_1 \circ \cdots \circ x_n \circ x_{n+2}$$

Ahora, supongamos que $M = \theta, N = \theta' \in \mathbf{CL}^{\mathrm{CB}}_{\mathcal{B}^{\mathbf{c}}_{\mathbf{C}}}$. De esta manera, la inducción sobre θ consiste en los siguientes casos:

- $\theta = \{\!|\, c_1 \mapsto M_1; \cdots ; c_n \mapsto M_n \,|\!\}$ $(n > 0)$

Si $\theta \to \theta'$, necesariamente sucederá que $M_j \to M'_j$ para algún $j = 1, \cdots, n$. A partir de la hipótesis inductiva, entonces, el resultado sigue de manera inmediata.

- $\theta = \phi \bullet M_1$

Este caso presenta cuatro subcasos, detallados a continuación.

(i) $\theta = \{\!|\, c_i \mapsto N_i \,|\!\}_{i=1}^n \bullet M_1 \overset{\textsc{DotCase}}{\to} \{\!|\, c_i \mapsto N_i\, M_1 \,|\!\}_{i=1}^n = \theta'$

$$\begin{aligned}
\theta_\lambda &= (\{\!|\, c_i \mapsto N_i \,|\!\}_{i=1}^n \bullet M_1)_\lambda \\
&= \{\!|\, c_i \mapsto N_{i\lambda} \,|\!\}_{i=1}^n \circ M_{1\lambda} \\
&= \{\!|\, c_i \mapsto N_{i\lambda}\, M_{1\lambda} \,|\!\}_{i=1}^n \\
&= \theta'_\lambda
\end{aligned}$$

En este caso, la reducción se da en cero pasos.

(ii) $\theta = \mathbf{B}^0\, \varphi \bullet P \bullet M_1 \overset{\text{B-Dist}}{\to} (\varphi \bullet M_1) \bullet (P\, M_1) = \theta'$
Debido a la Observación 3.6.1, debe existir $n \geq 0$, constructores $c_1, \cdots, c_n$ y términos en $\Lambda\mathcal{B}_c$ $N_1, \cdots, N_n$ tales que $\varphi_\lambda = \{\!|\, c_i \mapsto N_i \,|\!\}_{i=1}^n$. Entonces,

$$\begin{aligned}
\theta_\lambda &= (\mathbf{B}^0\, \varphi \bullet P \bullet M_1)_\lambda \\
&= (\underline{\lambda} x_1 x_2.\varphi_\lambda \circ x_2 \circ (x_1\, x_2)) \circ P_\lambda \circ M_{1\lambda} \\
&= (\underline{\lambda} x_1 x_2.\{\!|\, c_i \mapsto N_i \,|\!\}_{i=1}^n \circ x_2 \circ (x_1\, x_2)) \circ P_\lambda \circ M_{1\lambda} \\
&= \{\!|\, c_i \mapsto \lambda x_1 x_2.N_i\, x_2\, (x_1\, x_2) \,|\!\}_{i=1}^n \circ P_\lambda \circ M_{1\lambda} \\
&= \{\!|\, c_i \mapsto (\lambda x_1 x_2.N_i\, x_2\, (x_1\, x_2))\, P_\lambda\, M_{1\lambda} \,|\!\}_{i=1}^n \\
&\overset{\textsc{AppLam}}{\twoheadrightarrow} \{\!|\, c_i \mapsto N_i\, M_{1\lambda}\, (P_\lambda\, M_{1\lambda}) \,|\!\}_{i=1}^n \\
&= \{\!|\, c_i \mapsto N_i \,|\!\}_{i=1}^n \circ M_{1\lambda} \circ (P_\lambda\, M_{1\lambda}) \\
&= \theta'_\lambda
\end{aligned}$$

(iii) $\theta = \mathbf{B}^{n+1}\, \varphi \bullet M_1 \overset{\text{B-Dec}}{\to} \mathbf{B}^n\, (\varphi \bullet M_1) = \theta'$
Nuevamente por la Observación 3.6.1, debe existir $m \geq 0$, constructores $c_1, \cdots, c_m$

y términos en $\Lambda\mathcal{B}_c$ $N_1, \cdots, N_m$ tales que $\varphi_\lambda = \{\!|\, c_i \mapsto N_i \,|\!\}_{i=1}^m$. En consecuencia,

$$\begin{array}{rcl}
\theta_\lambda & = & (\mathbf{B}^{n+1}\,\varphi \bullet M_1)_\lambda \\
& = & (\underline{\lambda} x_1 \cdots x_{n+3}.\varphi_\lambda \circ x_1 \circ \cdots \circ x_{n+1} \circ x_{n+3} \circ (x_{n+2}\, x_{n+3})) \circ M_{1\lambda} \\
& = & (\underline{\lambda} x_1 \cdots x_{n+3}.\{\!|\, c_i \mapsto N_i \,|\!\}_{i=1}^m \circ x_1 \circ \cdots \circ x_{n+1} \circ x_{n+3} \circ (x_{n+2}\, x_{n+3})) \circ M_{1\lambda} \\
& = & \{\!|\, c_i \mapsto \lambda x_1 \cdots x_{n+3}.N_i\, x_1 \cdots x_{n+1}\, x_{n+3}\, (x_{n+2}\, x_{n+3}) \,|\!\}_{i=1}^m \circ M_{1\lambda} \\
& = & \{\!|\, c_i \mapsto (\lambda x_1 \cdots x_{n+3}.N_i\, x_1 \cdots x_{n+1}\, x_{n+3}\, (x_{n+2}\, x_{n+3}))\, M_{1\lambda} \,|\!\}_{i=1}^m \\
& \overset{\text{AppLam}}{\twoheadrightarrow} & \{\!|\, c_i \mapsto \lambda x_2 \cdots x_{n+3}.N_i\, M_{1\lambda}\, x_2 \cdots x_{n+1}\, x_{n+3}\, (x_{n+2}\, x_{n+3}) \,|\!\}_{i=1}^m \\
& = & \{\!|\, c_i \mapsto \lambda x_1 \cdots x_{n+2}.N_i\, M_{1\lambda}\, x_1 \cdots x_n\, x_{n+2}\, (x_{n+1}\, x_{n+2}) \,|\!\}_{i=1}^m \\
& = & (\underline{\lambda} x_1 \cdots x_{n+2}.\{\!|\, c_i \mapsto N_i \,|\!\}_{i=1}^m \circ M_{1\lambda} \circ x_1 \circ \cdots \circ x_n \circ x_{n+2} \circ (x_{n+1}\, x_{n+2})) \\
& = & (\underline{\lambda} x_1 \cdots x_{n+2}.\varphi_\lambda \circ M_{1\lambda} \circ x_1 \circ \cdots \circ x_n \circ x_{n+2} \circ (x_{n+1}\, x_{n+2})) \\
& = & \theta'_\lambda
\end{array}$$

(iv) Reducción interna
Inmediato a partir de la hipótesis inductiva, utilizando también las Observaciones 3.6.3 y 3.6.1 según dónde se dé la reducción.

- $\theta = \mathbf{B}^n\,\varphi$ $(n \in \mathbb{N})$

Este último caso sólo puede contener una reducción interna $\varphi \to \varphi'$, siendo $\varphi' \in \mathbf{CL}^{\mathrm{CB}}_{\mathcal{B}_\mathbf{c}}$ y en consecuencia $\theta' = \mathbf{B}^n\,\varphi'$. Luego,

$$\begin{array}{rcl}
\theta_\lambda & = & (\mathbf{B}^n\,\varphi)_\lambda \\
& = & \underline{\lambda} x_1 \cdots x_{n+2}.\varphi_\lambda \circ x_1 \circ \cdots \circ x_n \circ x_{n+2} \circ (x_{n+1}\, x_{n+2}) \\
& \overset{\text{HI}}{\twoheadrightarrow} & \underline{\lambda} x_1 \cdots x_{n+2}.\varphi'_\lambda \circ x_1 \circ \cdots \circ x_n \circ x_{n+2} \circ (x_{n+1}\, x_{n+2}) \\
& = & \theta'_\lambda
\end{array}$$

En forma análoga a lo expresado para el caso donde $M = \mathbf{A}^n\,\varphi$, el paso donde se utiliza la hipótesis inductiva también hace uso de la Observación 3.6.3. □

3.6.3. Propiedades

Las traducciones presentadas cumplen la propiedad de ser pseudo inversas, en el sentido de equivalencia y no de igualdad sintáctica. Esto es de relevancia en el marco del problema de unficación de orden superior, tal como se mencionó en la sección 3.1.1. De esta manera, dichas conversiones podrían tomarse como punto de partida al lidiar con aquel problema en un posible trabajo futuro.

Las proposiciones 3.6.2 y 3.6.3 muestran, respectivamente, cómo traducir un término (o case binder) en un sentido y luego traducir en el sentido inverso resulta en un nuevo término equivalente al original. Optamos por enunciar estos resultados sin detallar sus respectivas pruebas para no sobrecargar excesivamente el trabajo, desviando el foco de atención del mismo.

Proposición 3.6.2. Sea $M \in \mathbf{CL}_{\mathcal{B}_\mathbf{c}}$. Entonces,

$$(M_\lambda)_{\mathrm{CL}} =_{\mathrm{CL}_{\mathcal{B}_c}} M$$

Demostración Por inducción en M (omitida). □

Proposición 3.6.3. Sea $M \in \Lambda\mathcal{B}_c + \mathcal{B}$. Entonces,

$$(M_{\mathrm{CL}})_\lambda =_{\lambda\mathcal{B}_c} M$$

Demostración Por inducción en M, y utilizando el Lema 3.6.1 (omitida). □

El Lema aquí mencionado asegura que la traducción a $\lambda\mathcal{B}_c$ de un término generado por λ^* es equivalente a una abstracción usual, como es de esperar, con su cuerpo a su vez reconvertido a $\lambda\mathcal{B}_c$. Para el caso de case binders, tal abstracción es la abstracción simultánea que viene dada por el meta-operador $\underline{\lambda}$.

Lema 3.6.1. Sea $M \in \mathbf{CL}_{\mathcal{B}_\mathbf{c}}$. Entonces,

$$\begin{array}{rcl} M \in \mathbf{CL}^{\mathrm{T}}_{\mathcal{B}_\mathbf{c}} & \Rightarrow & (\lambda^* x.M)_\lambda =_{\lambda\mathcal{B}_c} \lambda x.M_\lambda \\ M = \theta \in \mathbf{CL}^{\mathrm{CB}}_{\mathcal{B}_\mathbf{c}} & \Rightarrow & (\lambda^* x.\theta)_\lambda =_{\lambda\mathcal{B}_c} \underline{\lambda} x.\theta_\lambda \end{array}$$

Demostración Por inducción en M (omitida). □

3.7. Extensionalidad

Resulta interesante preguntarse si en $\mathrm{CL}_{\mathcal{B}_c}$ existe alguna forma de extensionalidad, o si la extensionalidad en $\lambda\mathcal{B}_\mathcal{C}$ es bien traducida. En esta corta sección exponemos distintas formas que presenta la η-reducción tanto para términos como para case binders.

Proposición 3.7.1. Se tienen las siguientes propiedades:

1. Sea $\theta \in \mathcal{B}$ y $x \notin \mathrm{FV}(\theta)$. Entonces,

$$\underline{\lambda} x.\theta \circ x \quad \overset{\alpha}{\twoheadrightarrow}_{\textsc{LamApp}} \quad \theta \quad \text{con } \alpha = |\theta|$$

2. Sean $M, N \in \mathbf{CL}^{\mathrm{T}}_{\mathcal{B}_\mathbf{c}}$. Entonces,

$$(\lambda^* x.M)\, N \quad \twoheadrightarrow \quad M \quad \text{si } x \notin \mathrm{FV}(M)$$

3. Sea $\theta \in \mathbf{CL}^{\mathrm{CB}}_{\mathcal{B}_\mathbf{c}}$ y $N \in \mathbf{CL}^{\mathrm{T}}_{\mathcal{B}_\mathbf{c}}$. Entonces,

$$(\lambda^* x.\theta) \bullet N \quad \twoheadrightarrow \quad \theta \quad \text{si } x \notin \mathrm{FV}(\theta)$$

Demostración Consecuencia inmediata de la Proposición 3.5.2.

A partir de esto, y de la propiedad de abstracción de λ^*, se tiene:

Proposición 3.7.2.

1. Sean $M, N \in \mathbf{CL}^{\mathrm{T}}_{\mathcal{B}_\mathbf{c}}$. Entonces,

$$(\lambda^* x.M\, x)\, N \quad \twoheadrightarrow \quad MN \quad \text{si } x \notin \mathrm{FV}(M)$$

2. Sea $\theta \in \mathbf{CL}^{\mathrm{CB}}_{\mathcal{B}_\mathbf{c}}$ y $N \in \mathbf{CL}^{\mathrm{T}}_{\mathcal{B}_\mathbf{c}}$. Entonces,

$$(\lambda^* x.\theta \bullet x) \bullet N \quad \twoheadrightarrow \quad \theta \bullet N \quad \text{si } x \notin \mathrm{FV}(\theta)$$

A partir de esto último surgen reglas de simplificación para la representación de la abstracción en $\mathrm{CL}_{\mathcal{B}_c}$. Esto es, a la definición del operador λ^* (Figura 3.5) pueden agregarse o eliminarse las siguientes dos cláusulas independientemente, manteniendo los resultados presentados:

$$\begin{array}{lrcll} (\textsc{Const}) & \lambda^* x.Mx & = & M & (x \notin \mathrm{FV}(M)) \\ (\textsc{ConstCB}) & \lambda^* x.\theta \bullet x & = & \theta & (x \notin \mathrm{FV}(\theta)) \end{array}$$

La primera de estas cláusulas, ya incluida en la definición presentada, es la generalización de la situación de la regla adicional de conversión de Curry-Feys para CL clásica, mientras que la segunda expresa una situación análoga para case binders.

Observar que, de omitir Const, será preciso utilizar S-Intro.

3.8. Comentarios adicionales

El desarrollo de $\mathrm{CL}_{\mathcal{B}_c}$, en las primeras etapas, encontró trabas en relación a la confluencia del sistema. Una vez definida la sintaxis del cálculo (haciendo foco en la representación de la abstracción), el conjunto inicial de reglas de reducción vino dado por la unión de las reglas usuales de la lógica combinatoria más las reglas CaseApp y CaseCons de $\lambda\mathcal{B}_c$ y, por supuesto, nuevas reglas para tratar las construcciones introducidas de acuerdo a sus características[7]. Sin embargo, un análisis preliminar de los pares críticos permitió concluir que las reglas no eran lo suficientemente poderosas como para lograr cerrar ciertos pares lo cual, sin dudas, hace que el cálculo no sea confluente. Por ejemplo, ocurrían situaciones como la ilustrada por la Figura 3.8, en donde el diagrama allí presente no permitía cerrarse a partir de esta primera versión de las reglas.

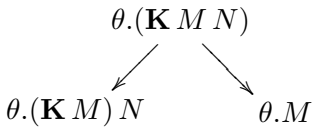

Figura 3.8: Par crítico no-cerrable a partir de las primeras reglas de $\mathrm{CL}_{\mathcal{B}_c}$

Siguiendo este ejemplo, se observó que, si uno generalizara y permitiera que $\theta.(\mathbf{K}\,M)\,N$ reduzca a $\theta.M$, el par crítico quedaría resuelto. No obstante, tal regla adicional introduciría un nuevo par que podría solucionarse de manera similar, y así sucesivamente. Este escenario motivó la compleción del sistema mediante esquemas de reglas basados en *clausuras vectoriales* de case binders. El problema inicial recae en la interacción entre la regla CaseApp y las reglas asociadas a $\mathbf{K}$, $\mathbf{S}$, $\mathbf{A}^0$ y $\mathbf{A}^{n+1}$. Similarmente, al seguir la metodología de Tait-Martin-Löf para asegurar la confluencia de la reducción, durante la prueba de satisfactibilidad de $\diamond$ surgieron inconvenientes que en última instancia fueron solucionados generalizando aún más dichos esquemas, introduciendo varios niveles de clausura vectorial. De esta manera, logramos finalmente obtener un conjunto de reglas satisfactorio, que es el que ilustra la Figura 3.1.

Desde un punto de vista implementativo, los vectores de case binders imponen la necesidad de explorar hacia adentro los términos para determinar si se puede o no aplicar alguno de los esquemas durante la evaluación.

[7]Por ejemplo, para el caso de $\mathbf{A}^0$, se tenía la regla $\mathbf{A}^0\theta\,M\,N \to (\theta \bullet N).(M\,N)$

Capítulo 4

Conclusiones

En esta tesis hemos estudiado distintas formulaciones de cálculos lambda con patrones desde la perspectiva otorgada por traducciones sintácticas entre ellos. Tales traducciones se vieron motivadas por múltiples propósitos, entre los cuales podemos mencionar la potencial transferencia de propiedades de un formalismo a otro, la oportunidad de dilucidar qué relaciones guardan entre ellos y la posibilidad de obtener implementaciones entendiendo la traducción como un proceso de compilación.

El trabajo expuesto se desarrolló en torno al cálculo $\lambda\mathcal{B}_c$. Esta tesis abordó el estudio de este cálculo, en tanto hemos propuesto traducciones hacia él y desde él, debido al interés que resulta del sistema para codificar funciones y patrones de manera unificada, tal como se expone en [3, 5].

Por un lado, presentamos una conversión de términos de $\lambda\mathcal{C}$, un cálculo con patrones de primer orden y abstracciones múltiples, a términos de $\lambda\mathcal{B}_c$, cuya característica esencial es la introducción de *case binders* (análisis de casos por constructores) y su particular interacción con las funciones. El cálculo de origen extiende la β-reducción clásica a abstracciones de múltiples patrones en donde un paso de reducción oculta el proceso de matching, que se realiza en forma atómica e instantánea. A diferencia de esto, $\lambda\mathcal{B}_c$ sólo dispone de abstracciones clásicas: el matching sobre patrones debe ser procesado secuencialmente a través de distintos case binders, que son propagados según las diversas reglas del cálculo. Tales reglas permiten, además, obtener un mayor aprovechamiento al evaluar ciertos términos cuyos equivalentes en $\lambda\mathcal{C}$ se encontrarían bloqueados debido a la fuerte restricción que impone la necesidad de instanciar patrones para poder reducir. Estas importantes diferencias impulsaron la búsqueda de una traducción entre ambos formalismos.

A partir de los resultados negativos de los primeros enfoques tomados, arribamos a una primera conclusión puramente argumentada de manera intuitiva: creemos firmemente que no es posible encontrar traducciones totales de $\lambda\mathcal{C}$ a $\lambda\mathcal{B}_c$. El mecanismo de análisis de casos de este último cálculo, que constituye la única herramienta para simular pattern matching y, en consecuencia, para simular las abstracciones múltiples, limita la posibilidad de distinguir entre patrones cuando éstos son tales que sus distintas posiciones (a excepción de la raíz) corresponden a variables en, por lo menos, uno de ellos. En tales escenarios, uno puede esperar un término arbitrario en las posiciones de variables correspondientes a las diferentes instancias de los patrones, lo cual deviene en la imposibilidad de generar un case binder que se comporte de manera acorde.

Para la traducción presentada demostramos no sólo que simula un paso de reducción β-multiple-pattern, sino también que lo hace en una cantidad lineal de pasos[1]. Además,

[1] Respecto del tamaño del patrón instanciado originalmente.

mostramos que también simula η-reducción en un paso y estudiamos la complejidad espacial en función de los tamaños de los términos y sus traducciones, probando que, en el peor caso, el tamaño de un término traducido guarda una relación lineal respecto del tamaño del término original. Estas buenas propiedades, entonces, refuerzan la posibilidad de analizar la conversión desde un punto de vista implementativo.

Además de esto, la experimentación con ciertas familias de términos nos hizo concluir que, bajo ciertas condiciones, los tamaños de los términos traducidos son estrictamente menores que los tamaños de los términos originales. Puntualmente, esto puede ocurrir cuando una abstracción múltiple contiene una gran cantidad de patrones que comparten parte de la estructura. Se observó que los case binders de $\lambda\mathcal{B}_c$ pueden aprovechar esta redundancia y disponer más eficientemente los datos.

En base al desarrollo de esta primera etapa, podemos delinear las siguientes posibilidades de trabajo a futuro:

- Proveer una extensión a $\lambda\mathcal{C}$ transfiriendo, en forma restringida, las reglas CaseLam y CaseApp de $\lambda\mathcal{B}_c$. Esto podría permitir una traducción inversa de $\lambda\mathcal{B}_c$ a $\lambda\mathcal{C}$.
- Estudiar sistemas de tipos para $\lambda\mathcal{C}$ y analizar de qué modo se traduce el tipado de este cálculo en relación al tipado de $\lambda\mathcal{B}_c$ propuesto en [55].
- Formalizar la intuición de que ninguna traducción puede ser total.

Por otra parte, se estudió un sistema de combinadores para $\lambda\mathcal{B}_c$ (llamado $\mathrm{CL}_{\mathcal{B}_c}$) en donde, tal como ocurre con la lógica combinatoria en relación al λ-cálculo clásico, el objetivo principal es eliminar el concepto de ligadores de variables con el fin de otorgar posibles implementaciones del cálculo original de manera más sencilla, sin necesidad de lidiar con las conocidas dificultades introducidas por los ligadores. Además de esto, la formulación de tal sistema se vio motivada, en mayor o menor grado, por una potencial transferencia de problemas sobre $\lambda\mathcal{B}_c$ a un contexto algebraico que permite, en muchos casos, un tratamiento más adecuado de los mismos (así ocurre, por ejemplo, con el problema de resolubilidad), y también, más concretamente, por la posibilidad de abordar el problema de unificación de orden superior sin la dificultad adicional de trabajar con variables ligadas. La traducción de $\lambda\mathcal{B}_c$ a $\mathrm{CL}_{\mathcal{B}_c}$, además de tener posibles usos futuros en relación a estas observaciones, permite estudiar el *gap* existente entre el cálculo y la implementación.

La prueba de confluencia de $\mathrm{CL}_{\mathcal{B}_c}$ no precisó de la técnica empleada en [5] (i.e., *divide-and-conquer*), quizás por la ausencia de ligadores. Sin embargo, la prueba desarrollada no permite obtener una nueva prueba de confluencia para $\lambda\mathcal{B}_c$, justamente porque reducción en éste no implica reducción en $\mathrm{CL}_{\mathcal{B}_c}$.

Es interesante notar el novedoso hecho de que esta formulación combinatoria permite tratar a los case binders casi como términos, en el sentido de que se ha definido para ellos aplicación y abstracción, acercándonos así, aunque sin serlo del todo, a los trabajos recientes de patrones como ciudadanos de primera clase.

Las posibilidades de trabajo a futuro incluyen el estudio de la noción de reducción fuerte (cf. [33]) en $\mathrm{CL}_{\mathcal{B}_c}$.

La experiencia adquirida a lo largo del trabajo permite concluir que las traducciones sintácticas constituyen un mecanismo interesante no sólo para estudiar un cálculo a través de otro sino además como herramienta para comparar distintos aspectos entre formalismos de naturalezas opuestas. En este caso puntual, las traducciones desarrolladas fueron de utilidad para estudiar la representación y el tratamiento de los patrones en distintos cálculos. En

virtud de estas diferencias, pudimos comprobar comportamientos muy distintos entre uno y otro formalismo.

Apéndice A

Implementación

En forma complementaria al desarollo teórico, se han implementado en `Haskell` [61] los distintos cálculos estudiados ($\lambda\mathcal{C}$, $\lambda\mathcal{B}_c$ y $\mathrm{CL}_{\mathcal{B}_c}$) y las respectivas traducciones. A continuación describiremos a alto nivel la disposición de los módulos involucrados y las funcionalidades que cada uno provee.

Cada cálculo se ha implementado en forma individual a través de una serie de módulos, y cada uno de estos módulos queda completamente definido por un archivo que lleva su mismo nombre con la extensión `hs`. La arquitectura general es común a los tres cálculos. Siendo `X` cualquiera de ellos, los módulos se organizan de la siguiente manera:

- `XTypes`: declara los tipos de datos necesarios (términos, y además case binders para el caso de $\mathrm{CL}_{\mathcal{B}_c}$) e instacia la clase `Show` para permitir visualizar los términos. También define ciertas funciones auxiliares (variables libres, tamaño, etc.).
- `XParser`: contiene el código generado en forma automática por `Happy` [26], a partir de la gramática especificada en el archivo `XParser.y`. Este módulo provee un lexer y un parser para el cálculo en cuestión, permitiendo así manipular el cálculo a través de cadenas de caracteres.
- `X`: se trata del módulo principal del cálculo. Allí se definen las funciones de reducción en un paso (que implementa, desde ya, cada una de las reglas), reducción en N pasos y normalización [1]. La estrategia de reducción implementada es *leftmost outermost* [2]. Se optó por ésta debido a que resulta estándar en el λ-cálculo clásico, lo que garantiza normalización en caso de ser posible. Tanto en $\lambda\mathcal{B}_c$ como en $\lambda\mathcal{C}$ no existen, hasta donde sabemos, resultados de estandarización; no obstante, se estima que esta estrategia podría sentar las bases para una primera posibilidad de estandarización en dichos cálculos.
- `XExamples`: define un conjunto de ejemplos y casos de prueba para las reglas de reducción del cálculo. Estos ejemplos involucran funciones recursivas (mediante combinadores de punto fijo), números naturales y listas (definidos como tipos algebraicos).

Los módulos que implementan las traducciones son independientes a cada cálculo individual:

[1] La función de normalización simplemente reduce tanto como puede. Por supuesto, no terminará si el argumento no posee forma normal.

[2] A excepción de algunos casos puntuales como, por ejemplo, la aplicación de una abstracción múltiple a un término cualquiera en $\lambda\mathcal{C}$. En tal situación, se reducirá primero el argumento hasta el punto en el que instancie alguno de los patrones presentes en la abstracción múltiple. Una vez hecho esto, se reducirá la aplicación.

- `Translator`: implementa la traducción de $\lambda\mathcal{C}$ a $\lambda\mathcal{B}_c$. Existe además un módulo llamado `TranslatorChecker` que define funciones auxiliares para verificar la conmutación de la reducción y la traducción para un término dado como argumento y para comparar los tamaños del argumento y de su respectiva traducción.
- `MapsLBc-CLBc`: implementa los mapeos entre $\lambda\mathcal{B}_c$ y $\mathrm{CL}_{\mathcal{B}_c}$. Para ello, se vale de una función que implementa λ^*. También provee un conjunto de casos de prueba; algunos de ellos consisten en aplicar todas las traducciones implementadas a un término $\lambda\mathcal{C}$ y luego normalizar, para analizar si el resultado arrojado es el esperado.

Por último, listamos los nombres de las funciones de mayor relevancia y una breve descripción de las mismas. Por simplicidad, asumiremos que `a` es cualquiera de los tipos de datos `LambdaCTerm`, `LambdaBcTerm` o `CLBcTerm`:

- `reduce':: a -> Maybe a`: reducción en un paso. El valor de retorno es `Nothing` sólo si el argumento es forma normal.
- `normalize':: a -> (Integer, a)`: normalización. El par resultante indica la cantidad de reducciones efectuadas y la forma normal obtenida.
- `translate':: Int -> LambdaCTerm -> LambdaBcTerm`: implementación de la función Ψ. El primer argumento es una variable interna para garantizar que las variables $v \in \mathfrak{V}$ son frescas. Sólo tiene incidencia en nombres de variables ligadas.
- `toCLBc':: LambdaBcTerm -> CLBcTerm`: la traducción de $\lambda\mathcal{B}_c$ a $\mathrm{CL}_{\mathcal{B}_c}$.
- `toLBc':: CLBcTerm -> LambdaBcTerm`: la traducción de $\mathrm{CL}_{\mathcal{B}_c}$ a $\lambda\mathcal{B}_c$.

Tomamos la convención de que los nombres de funciones con tilde trabajan sobre los términos propiamente dichos. Por otro lado, estos mismos nombres sin tilde denotarán funciones que esperan argumentos de tipo `String`, i.e., cadenas de caracteres que serán interpretadas para generar el término que representan. Así, por ejemplo, `reduce :: String -> Maybe a`.

Esta implementación está disponible en

`http://www-2.dc.uba.ar/materias/reescritura/lsanti/`

Bibliografía

[1] M. Abadi, L. Cardelli, P.-L. Curien, J.-J. Lévy. *Explicit Substitutions.* Journal of Functional Programming 4(1), 1991.

[2] T. Arts, J. Giesl. *Termination of term rewriting using dependency pairs.* Theoretical Computer Science, 236, 2000.

[3] A. Arbiser. *Explicit substitution systems and subsystems.* Tesis de Doctorado, Universidad de Buenos Aires, 2005.

[4] A. Arbiser. *On the expressive power of calculi of explicit substitution.* Versión ampliada. Perspectives in Universal Logic, pp. 159-177. Polimetrica. Julio 2007.

[5] A. Arbiser, A. Miquel, A. Ríos. *A Lambda-Calculus with constructors.* 17th International Conference on Rewriting Techniques and Applications. Seattle, agosto de 2006. Lecture Notes in Computer Science, vol. 4098, pp. 181-196. Springer-Verlag ISBN: 978-3-540-36834-2 ISSN: 0302-9743

[6] A. Arbiser, A. Miquel, A. Ríos. *The lambda calculus with constructors: syntax, confluence and separation.* J. of Functional Programming, 2008 (a aparecer).

[7] A. Asperti, G. Longo. *Categories, types and structures. Category theory for the working computer scientist.* MIT Press. Cambridge, 1991.

[8] F. Baader, T. Nipkow. *Term Rewriting and All That.* Cambridge University Press, 1998.

[9] H.P. Barendregt. *The Lambda Calculus: its Syntax and Semantics.* Studies in Logic and the Foundations of Mathematics 103. North-Holland, Amsterdam, revised edition, 1984.

[10] H.P. Barendregt. *Lambda Calculi with Types.* Handbook of Logic in Computer Science, Vol II, Chap 2. Clarendon Press, Oxford, 1992.

[11] G. Barthe, H. Cirstea, C. Kirchner, L. Liquori. *Pure pattern type systems.* Proc. POPL'03, New Orleans, USA, January 2003. ftp://ftp-sop.inria.fr/everest/personnel/Gilles.Barthe/popl03.ps.gz

[12] L. Bachmair, N. Dershowitz. *Critical pair criteria for completion.* Journal of Symbolic Computation, 6(1), 1988.

[13] M. Bezem, J. W. Klop, R. de Vrijer (eds). *Term Rewriting Systems (TeReSe).* Cambridge University Press, 2003.

[14] F. Blanqui. *Type Theory and Rewriting.* Thèse de Doctorat. Université Paris XI. 2001.

[15] S. Cerrito, D. Kesner. *Pattern Matching as Cut Elimination.* Theoretical Computer Science. 323:71-127, 2004.

[16] H. Cirstea, C. Kirchner. *Rho-calculus, the rewriting calculus.* CCL 1998.

[17] H. Cirstea, C. Kirchner. *The rewriting calculus - Part I.* L. J. of the IGPL, 9(3). Oxford University Press, 2001.

[18] A. Church, J.B. Rosser. *Some properties of conversion.* Transactions of the American Mathematical Society 39, 1936.

[19] A. Church. *The calculi of lambda-conversion.* Annals of Mathematical Studies vol. 6. Princeton, 1941.

[20] H. B. Curry. *Functionality in combinatory logic.* Proc. Nat. Acad. Science USA 20, 584-590, 1934.

[21] H. B. Curry, R. Feys. *Combinatory Logic, Vol I.* Studies in Logic and the Foundation of Mathematics, North Holland 1958.

[22] H. B. Curry. *Modified basic functionality in combinatory logic.* Dialectica 23, 83-92, 1969.

[23] H. B. Curry, J. R. Hindley, J. P. Seldin. *Combinatory Logic, Vol II.* Studies in Logic and the Foundation of Mathematics, North Holland 1972.

[24] N. Dershowitz. *Orderings for term rewriting systems.* Theoretical Computer Science, 17(3), 1982.

[25] D. J. Dougherty. *Higher-Order Unification via Combinators.* Dept. of Mathematics, Wesleyan University, USA, 1993.

[26] A. Gill, S. Marlow. *Happy: The Parser Generator for Haskell.* `http://haskell.org/happy/`.

[27] J.-Y. Girard. *Locus solum: From the rules of logic to the logic of rules.* Mathematical Structures in Computer Science, 11(3):301–506, 2001.

[28] G. Huet. *A unification algorithm for typed -calculus.* Theoretical ComputerScience, 1:27–57, 1975.

[29] B. Jay. *The pattern calculus.* ACM TOPLAS, 26(6):911-937, 2004.

[30] B. Jay. *Typing the pattern calculus.* Proc. HOR 2006.

[31] B. Jay. *Typing First-class patterns.* Proc. HOR 2006.

[32] B. Jay, D. Kesner. *Pure pattern calculus.* University of Technology, Sydney, and PPS, CNRS and Université Paris 7, 2007.

[33] D. J. Dougherty, P. Johann. *A Combinator Logic Approach to Higher-Order E-unification.* Dept. of Mathematics, Wesleyan University, and Dept. of Math. and Computer Science, Hobart and Williams Smith College, USA, 1995.

[34] J.-P. Jouannaud, M.Okada. *Abstract Data Type Systems.* TCS 173:349-391, 1997.

[35] J.-P. Jouannaud, A. Rubio. *The higher-order recursive path ordering.* 14th Annual IEEE Symposium on Logic in Computer Science (LICS), IEEE Computer Society Press, 1999.

[36] W. Kahl. *Basic Pattern Matching Calculi: Syntax, Reduction, Confluence, and Normalisation.* Technical Report 16 McMaster University, 2003.

[37] W. Kahl. *Basic Pattern Matching Calculi: A Fresh View on Matching Failure*. In Y. Kameyama and P. Stuckey, editors, *Functional and Logic Programming, Proceedings of FLOPS 2004*, volume 2998 of *LNCS*, pages 276–290. Springer, 2004.

[38] S. Kamin, J.-J. Lévy. *Attempts for generalizing the recursive path orderings.* University of Illinois, 1980.

[39] Z. Khasidashvili. *Expression reduction Systems.* Proc. IN Vekua Institute of Applied Mathematics, vol. 36, Tbilisi, 1990.

[40] D. Kesner. *Confluence properties of extensional and non-extensional lambda-calculi with explicit substitutions.* Rewriting Techniques and Applications RTA96, LNCS 1103, 1996.

[41] D. Kesner, L. Puel, V. Tannen. *A Typed Pattern Calculus.* CNRS, Université Paris-Sud, 1998.

[42] D. Kesner. *Confluence of extensional and non-extensional lambda-calculi with explicit substitutions.* Theoretical Computer Science, Vol. 238, n. 1-2, 2000.

[43] D. Kesner, B.Jay. *Pure pattern calculus.* ESOP 2006. LNCS 3924.

[44] J.-W. Klop. *Combinatory Reduction Systems.* Vol. 127 of Mathematical Centre Tracts. CWI, Amsterdam. PhD Thesis, 1980.

[45] J. W. Klop, V. van Oostrom, R. de Vrijer. *Lambda Calculus with Patterns.* Theoretical Computer Science, 398(1-3):16-31, 2008.

[46] D. E. Knuth, P. B. Bendix. *Simple Word Problems in Universal Algebras.* In J. Leech, ed., Computational Problems in Abstract Algebra, 263–297. Pergamon Press, Oxford, 1970.

[47] S. Lucas. *Reescritura con Restricciones de Reemplazamiento.* Tesis de doctorado, Univ. Pol. Valencia, 1998.

[48] G. Manzonetto. *Models and theories of λ-calculus.* PhD. thesis, Universitá Ca'Foscari di Venezia, 2008.

[49] A. Martelli, U. Montanari. *An efficient unification algorithm.* ACM Transactions on Programming Languages and Systems, pages 258-282, Vol. 4, No. 2, 1982.

[50] P.-A. Melliès. *Description Abstraite des Systèmes de Réécriture.* Thèse de Doctorat, Université Paris VII, 1996.

[51] R. Milner, M. Tofte, and R. Harper. *The definition of Standard ML.* MIT Press, 1990.

[52] T. Nipkow. *A Critical Pair Lemma for Higher-Order Rewrite Systems.* First Annual Workshop on Logical Frameworks. 1990.

[53] T. Nipkow. *Higher order critical pairs.* 6th Anual IEEE Symposium on Logic in Computer Science (LICS), IEEE Computer Society Press, 1991.

[54] *The Objective Caml language*. `http://caml.inria.fr/`.

[55] B. Petit. *A polymorphic type system for lambda-calculus with constructors.* 9th. International Conference on Typed Lambda Calculi and Applications, Brasil, 2009.

[56] V. van Oostrom, F. van Raamsdonk. *Comparing Combinatory Reduction Systems and Higher-Order Rewrite Systems.* Proc. 1st International Workshop on Higher-Order Algebra, Logic and Rewriting (HOA93), LNCS 618, Springer Verlag, 1993.

[57] V. van Oostrom, F. van Raamsdonk. *Weak orthogonality implies confluence: the higher order case.* 3rd International Symposium on Logical foundations of Computer Science, LNCS 813, Springer Verlag, 1994.

[58] V. van Oostrom. *Lambda Calculus with Patterns.* Technical report, Vrijke Universiteit, 1990.

[59] B. Pagano. *Des calculs de substitution explicite et de leur application à la compilation des langages fonctionnels.* Thèse de Doctorat, Université Pierre et Marie Curie, 1998.

[60] S. Peyton Jones. *The Implementation of Functional Programming Languages.* Prentice Hall, 1987.

[61] S. Peyton Jones et al. *The Revised Haskell 98 Report.* Cambridge Univ. Press, 2003. También en http://haskell.org/.

[62] F. van Raamsdonk. *Confluence and Normalization for Higher-Order rewriting.* PhD Thesis, Amsterdam University, 1996.

[63] M. Schönfinkel. *Über die Bausteinen der matematische Logik.* Math. Ann. 92:305-316, 1924.

[64] P. Selinger. *The Lambda Calculus is Algebraic.* Department of Mathematics and Statistics, University of Ottawa, Canada.

[65] M. H. B. Sörensen, P. Urzyczyn. *Lectures on the Curry-Howard Isomorphism.* University of Copenhagen/University of Warsaw, 1998.

[66] D. A. Turner. *A new implementation technique for applicative languages.* Software - Practice and Experience, 9:31–49, 1979.

[67] D. A. Turner. *Another Algorithm for Bracket Abstraction.* Journal of Symbolic Logic, 44(2):267–270, 1979.

[68] B. Wack, C. Houtmann. *Strong Normalization in two Pure Pattern Type Systems.* Université Henri Poincaré. LORIA, Francia, 2007.